PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

ALGORITMO CUCKOO SEARCH PARA LA PLANIFICACIÓN DE CITAS DE RECEPCIÓN DE MATERIA PRIMA EN EMPRESAS DE PRODUCCIÓN MULTIPLANTA

TESIS PARA OPTAR POR EL TÍTULO PROFESIONAL DE INGENIERO INFORMÁTICO

AUTOR

Steven Alonso Labajos Trigoso

ASESOR

Mg. Rony Cueva Moscoso

Lima, marzo, 2021

Resumen

En la actualidad, una planificación adecuada de citas de recepción de materia prima es un factor clave en las empresas productoras, ya que se debe garantizar la disponibilidad de los insumos cuando los procesos productivos los requieran. De lo contrario, se corre el riesgo de no poder satisfacer la demanda y de obstaculizar las operaciones de la empresa, generando un incremento en los costos.

Dada la alta complejidad involucrada en realizar una planificación de citas de recepción, debido al número de variables y restricciones que utiliza, este tipo de problema se clasifica como NP-difícil, por lo que resolverlo mediante técnicas determinísticas o matemáticas resulta muy complejo. Por ello, para resolver este tipo de problemas se utilizan técnicas metaheurísticas, las cuales proveen algoritmos eficientes y prácticos que permiten obtener soluciones de calidad en un tiempo razonable.

En este proyecto de tesis se utiliza el algoritmo Cuckoo Search para resolver el problema de planificación de citas de recepción de materia prima en empresas de producción multiplanta, ya que, a pesar de su desarrollo reciente, está demostrando tener excelentes resultados al resolver problemas de optimización, debido a su rápida convergencia y robusta búsqueda global. Además, no se ha encontrado en la literatura una aplicación del algoritmo mencionado para el problema que se desea resolver.

Se desarrolla también una aplicación del algoritmo genético para el caso estudiado, siendo este uno de los más representativos y populares para resolver problemas de optimización, con la finalidad de validar la eficiencia del algoritmo Cuckoo Search respecto a uno de los más usados en la actualidad.

Índice

Índice de tablas

Índice de figuras

Capítulo 1. Generalidades

1.1 Problemática

A continuación, se presenta el contexto de la problemática, así como las principales causas y consecuencias de la misma, con la finalidad de evidenciar claramente la necesidad identificada y respaldar la relevancia del proyecto de fin de carrera.

1.1.1 Árbol de Problemas

Tabla 1. Árbol de problemas

Árbol de problemas				
Problemas efectos	Inconsistencias diversas en la planificación realizada, debido a la elaboración manual de soluciones.	Falta de abastecimiento en la producción y sobrecostos para la empresa debido a una mala planificación.	Escasas aplicaciones informáticas brindan el nivel de soporte requerido en resolver una planificación de citas de recepción.	Sobrecostos para la empresa debido a la necesidad de realizar ajustes posteriores para corregir inconsistencias en la planificación.
Problema central	La planificación de citas presenta inconsistencias en la programación de horarios.			
Problemas causas	El personal encargado, al no tener herramientas de apoyo adecuadas, suele realizar la planificación de citas de recepción de forma manual.	No se consideran parámetros y restricciones suficientes en los sistemas actuales como para generar una planificación	Alto nivel de complejidad y consumo de recursos computacionales para resolver problemas de planificación de citas, clasificados	Las soluciones actuales no tienen los métodos adecuados para realizar una planificación de citas, ya que se basan

		útil de citas de recepción.	como NP-difíciles, especialmente en productoras multiplanta.	exclusivamente en la necesidad de la materia prima y no consideran la capacidad de almacenamiento

Fuente: Elaboración propia.

1.1.2 Descripción

Las condiciones altamente competitivas en las que actualmente se desarrollan los negocios han provocado que las compañías busquen mayores oportunidades y alternativas que les permitan un mejor posicionamiento en los mercados globales. Por ello, en las últimas décadas, la logística se ha convertido en un elemento estratégico fundamental, repleto de avances tecnológicos, alta competencia y mayor exigencia de parte del cliente (García, 2016).

La logística se encarga de administrar los bienes de una forma eficiente, siendo uno de sus enfoques principales el abastecimiento de la empresa (Anca, 2019). Gestionar el mismo se asocia a un problema de toma de decisiones en el cual una de las variables más significativas es ¿cuándo pedir?, ya que, con una planificación inadecuada, se corre el riesgo de no poder satisfacer la demanda y de obstaculizar las operaciones de la empresa. Por lo tanto, en este sector, las decisiones de optimización son muy frecuentes y cada vez más complejas (Pérez et al., 2013).

En este escenario, la complejidad aumenta al tratar con empresas productoras de tipo multiplanta, dado que estas comparten los recursos entre todas las pertenecientes a la red de producción, por lo que realizar una planificación de abastecimiento no sólo considera las características propias de la planta que lo solicita, sino que es posible destinar el

almacenamiento temporal de la materia prima a otra ubicación hasta que sea requerida en el destino original (Alvarez, 2007).

Tradicionalmente, el personal encargado de la planificación del abastecimiento realiza esta actividad de forma manual, invirtiendo una gran cantidad de esfuerzo y tiempo en el estudio y ajuste de soluciones aceptables. Esta manera de trabajar no garantiza la optimalidad de la solución e incurre en diversas inconsistencias (Ruescas, 2016). Una planificación inconsistente de las citas de abastecimiento manifiesta una ineficiente gestión de los elementos logísticos relacionados, generando en la empresa un incremento en los costos, ya sea por almacenamiento excesivo o en el transporte recurrente de materia prima escasa (Malindzakova, 2019). Por lo tanto, resulta necesario considerar factores clave como la capacidad de almacenamiento y la prioridad de recepción entre estas.

Los sistemas ERP, una de las principales propuestas de solución para la planificación de recursos y gestión de la información, ha mostrado diversas desventajas para alinear el uso de los mismos con las necesidades propias de las empresas (Banaeianjahromi, 2016). Los módulos relacionados que ofrece son muy limitados para proveer un soporte adecuado a la planificación; es decir, pueden identificar las necesidades de materia prima y programar órdenes de compra; sin embargo, esta planificación no considera la información suficiente para que resulte útil al usuario, generando, en muchos casos, programaciones inadecuadas que requieren ajustes posteriores (de Man & Strandhagen, 2018). Esto puede perjudicar severamente a los procesos de producción, ya que no se garantiza la disponibilidad de la materia prima en las fechas requeridas debido al bajo nivel de parametrización de las soluciones (de Man & Strandhagen, 2018).

Dada la alta complejidad involucrada en realizar una planificación de citas de recepción, este tipo de problema se clasifica como NP-difícil y, por el tipo de variables y de restricciones que maneja, de optimización combinatoria, donde un incremento en el tamaño del problema

produce un incremento exponencial del espacio de soluciones, por lo que resolverlo mediante técnicas determinísticas o matemáticas resulta muy complicado, ya que demanda un alto consumo de recursos computacionales (Pandolfi et al., 2018).

Para resolver este tipo de problemas, en los cuales resulta inviable recorrer todo el espacio de combinaciones posibles, se utilizan técnicas metaheurísticas, las cuales proveen algoritmos eficientes y prácticos que permiten obtener soluciones de calidad en un tiempo razonable (Yang, 2010), mediante estrategias de alto nivel para explorar espacios de búsqueda de forma guiada (Blum & Roli, 2003).

Uno de los algoritmos más populares y estudiados es el algoritmo genético, el cual ha sido aplicado en diversos tipos de problemas, ya sea en su forma base o modificado con la finalidad de obtener un mejor rendimiento. Este algoritmo ha demostrado un gran nivel de adaptabilidad a diferentes objetivos brindando resultados satisfactorios (Yang, 2010).

Por otro lado, un algoritmo novedoso y presentado en el 2009 está demostrando tener excelentes resultados al resolver problemas de optimización, debido a su rápida convergencia y robusta capacidad de búsqueda global (Maadi, Javidnia & Ramezani, 2018). Este algoritmo se denomina Cuckoo Search y está basado en la estrategia de reproducción de los Cuckoos, los cuales depositan sus huevos en nidos de otras aves, con la finalidad de que estos se reproduzcan como si fuesen propios del anfitrión. El objetivo del algoritmo es encontrar la región del espacio donde la mayoría de estos sobreviven, la cual representa a la mejor solución encontrada. (Maadi, Javidnia & Ramezani, 2018).

Por lo expuesto, en este proyecto de tesis se utilizará el algoritmo Cuckoo Search para resolver el problema de planificación de citas de recepción de materia prima en empresas de producción multiplanta. Además, se ha seleccionado al algoritmo genético, uno de los más representativos, como elemento de comparación respecto al propuesto.

1.1.3 Problema seleccionado

El problema seleccionado es la planificación de citas de recepción de materia prima en una empresa productora de tipo multiplanta. El enfoque está relacionado directamente a evitar la falta de materiales requeridos para la producción, considerando factores relevantes en la logística de abastecimiento como la capacidad de almacenamiento disponible y la priorización entre necesidades. Se propone el uso del algoritmo Cuckoo Search para resolver este problema, el cual será adaptado al contexto presentado con la finalidad de realizar una planificación útil para el usuario y la empresa.

1.2 Objetivos

1.2.1 Objetivo general

Implementar el algoritmo Cuckoo Search para realizar la planificación de citas de recepción de materia prima en una empresa productora multiplanta.

1.2.2 Objetivos específicos

O1. Definir los parámetros, restricciones y cómo se relacionan mediante la función objetivo, la cual sirve para realizar la evaluación de posibles soluciones.

O2. Adaptar el algoritmo genético para resolver el problema de planificación.

O3. Diseñar el algoritmo Cuckoo Search para resolver el problema de planificación.

O4. Desarrollar un software para la ejecución de los algoritmos Cuckoo Search y genético.

O5. Desarrollar la comparación de resultados entre los algoritmos Cuckoo Search y genético.

1.2.3 Resultados esperados

R1.1 Lista de definición de parámetros y restricciones para el problema de planificación.

R1.2 Formulación de la función objetivo para el problema de planificación.

R2.1 Diseño de estructuras de datos que sirven de soporte para el algoritmo genético adaptado al problema de planificación.

R2.2 Algoritmo genético adaptado al problema de planificación.

R3.1 Diseño de estructuras de datos que sirven de soporte para el algoritmo Cuckoo Search adaptado al problema de planificación.

R3.2 Algoritmo Cuckoo Search diseñado para resolver el problema de planificación.

R4.1 Codificación del algoritmo genético adaptado al problema de planificación.

R4.2 Codificación del algoritmo Cuckoo Search diseñado para resolver el problema de planificación.

R4.3 Interfaz gráfica para la ejecución de los algoritmos.

R5.1 Calibración de variables.

R5.2 Informe de experimentación numérica.

1.2.4 Mapeo de objetivos, resultados y verificación

Tabla 2. Medios de verificación de los resultados esperados del primer objetivo específico.

Objetivo 1: Definir los parámetros, restricciones y cómo se relacionan mediante la función objetivo, la cual sirve para realizar la evaluación de posibles soluciones.		
Resultado	**Medio de verificación**	**Indicador objetivamente verificable**
R1.1 Lista de definición de parámetros y restricciones para el problema de planificación.	Documento que contiene la definición, descripción y formulación de los parámetros y restricciones.	Aprobación del documento por parte del especialista en logística.
R1.2 Formulación de la función objetivo para el problema de planificación.	Documento que contiene la definición y formulación de la función objetivo.	Aprobación del documento por parte del especialista en algoritmia.

Fuente: Elaboración propia.

Tabla 3. Medios de verificación de los resultados esperados del segundo objetivo específico.

Objetivo 2: Adaptar el algoritmo genético para resolver el problema de planificación.		
Resultado	**Medio de verificación**	**Indicador objetivamente verificable**
R2.1 Diseño de estructuras de datos que sirven de soporte para el algoritmo genético adaptado al problema de planificación.	Documento que contiene la definición de las estructuras bio-inspiradas utilizadas en el algoritmo genético.	Aprobación del documento por parte del especialista en algoritmia.
R2.2 Algoritmo genético adaptado al problema de planificación.	Documento que contiene el pseudocódigo del algoritmo genético adaptado al problema.	-Aprobación del diseño por parte del especialista en algoritmia. -Pruebas de caja blanca sobre la base de resultados conocidos.

Fuente: Elaboración propia.

Tabla 4. Medios de verificación de los resultados esperados del tercer objetivo específico.

Objetivo 3: Diseñar el algoritmo Cuckoo Search para resolver el problema de planificación.		
Resultado	**Medio de verificación**	**Indicador objetivamente verificable**
R3.1 Diseño de estructuras de datos que sirven de soporte para el algoritmo Cuckoo Search adaptado al problema de planificación.	Documento que contiene la definición de las estructuras bio-inspiradas utilizadas en el algoritmo Cuckoo Search.	Aprobación del documento por parte del especialista en algoritmia.
R3.2 Algoritmo Cuckoo Search diseñado para resolver el problema de planificación.	Documento que contiene el pseudocódigo del algoritmo Cuckoo Search	-Aprobación del diseño por parte del especialista en algoritmia.

	diseñado para resolver el problema de planificación.	-Pruebas de caja blanca sobre la base de resultados conocidos.

Fuente: Elaboración propia.

Tabla 5. Medios de verificación de los resultados esperados del cuarto objetivo específico.

Objetivo 4: Desarrollar un software para la ejecución de los algoritmos Cuckoo Search y genético.		
Resultado	**Medio de verificación**	**Indicador objetivamente verificable**
R4.1 Codificación del algoritmo genético adaptado al problema de planificación.	Código fuente del algoritmo genético.	Pruebas unitarias a los métodos utilizados en el algoritmo.
R4.2 Codificación del algoritmo Cuckoo Search diseñado para resolver el problema de planificación.	Código fuente del algoritmo Cuckoo Search.	Pruebas unitarias a los métodos utilizados en el algoritmo.
R4.3 Interfaz gráfica para la ejecución de los algoritmos.	Código fuente y GUI de la interfaz gráfica.	Pruebas de integración entre el algoritmo y la interfaz.

Fuente: Elaboración propia.

Tabla 6. Medios de verificación de los resultados esperados del quinto objetivo específico.

Objetivo 5: Desarrollar la comparación de resultados entre los algoritmos Cuckoo Search y genético.		
Resultado	**Medio de verificación**	**Indicador objetivamente verificable**

R5.1 Calibración de variables.	Documento que contiene la calibración de variables para la ejecución de los algoritmos.	Aprobación del documento por parte del especialista en algoritmia.
R5.2 Informe de experimentación numérica.	Informe que contiene el análisis de resultados de la experimentación numérica realizada.	Aprobación del documento por parte del especialista en algoritmia.

Fuente: Elaboración propia.

1.3 Herramientas y métodos

Tabla 7. Herramientas y métodos a utilizar para la obtención de resultados del primer objetivo específico.

Objetivo 1: Definir los parámetros, restricciones y cómo se relacionan mediante la función objetivo, la cual sirve para realizar la evaluación de posibles soluciones.	
Resultado	**Herramientas y métodos**
R1.1 Lista de definición de parámetros y restricciones para el problema de planificación.	No aplica.
R1.2 Formulación de la función objetivo para el problema de planificación.	No aplica.

Fuente: Elaboración propia.

Tabla 8. Herramientas y métodos a utilizar para la obtención de resultados del segundo objetivo específico.

Objetivo 2: Adaptar el algoritmo genético para resolver el problema de planificación.	
Resultado	**Herramientas y métodos**
R2.1 Diseño de estructuras de datos que sirven de soporte para el algoritmo genético adaptado al problema de planificación.	No aplica.
R2.2 Algoritmo genético adaptado al problema de planificación.	Pseudocódigo.

Fuente: Elaboración propia.

Tabla 9. Herramientas y métodos a utilizar para la obtención de resultados del tercer objetivo específico.

Objetivo 3: Diseñar el algoritmo Cuckoo Search para resolver el problema de planificación.	
Resultado	**Herramientas y métodos**
R3.1 Diseño de estructuras de datos que sirven de soporte para el algoritmo Cuckoo Search adaptado al problema de planificación.	No aplica.
R3.2 Algoritmo Cuckoo Search diseñado para resolver el problema de planificación.	Pseudocódigo.

Fuente: Elaboración propia.

Tabla 10. Herramientas y métodos a utilizar para la obtención de resultados del cuarto objetivo específico.

Objetivo 4: Desarrollar un software para la ejecución de los algoritmos Cuckoo Search y genético.	
Resultado	**Herramientas y métodos**
R4.1 Codificación del algoritmo genético adaptado al problema de planificación.	Netbeans Java Git Algunas buenas prácticas de Scrum
R4.2 Codificación del algoritmo Cuckoo Search diseñado para resolver el problema de planificación.	Netbeans Java Git Algunas buenas prácticas de Scrum
R4.3 Interfaz gráfica para la ejecución de los algoritmos.	Netbeans Java Git Algunas buenas prácticas de Scrum

Fuente: Elaboración propia.

Tabla 11. Herramientas y métodos a utilizar para la obtención de resultados del quinto objetivo específico.

Objetivo 5: Desarrollar la comparación de resultados entre los algoritmos Cuckoo Search y genético.	
Resultado	**Herramientas y métodos**
R5.1 Calibración de variables.	Netbeans Java RStudio
R5.2 Informe de experimentación numérica.	RStudio ANOVA Pruebas estadísticas no paramétricas

Fuente: Elaboración propia.

1.3.1 Pseudocódigo

El pseudocódigo es una herramienta de programación en el que las instrucciones se escriben en palabras similares al inglés o español, las cuales facilitan tanto la escritura como la lectura de programas. En esencia, el pseudocódigo se puede definir como un lenguaje de especificaciones de algoritmos (Gómez, 2012).

Aunque no existen reglas para escritura del pseudocódigo es español, se utilizan palabras reservadas básicas, estas palabras son traducción libre de palabras reservadas de lenguajes como C y Pascal (Gómez, 2012).

Se utilizará en el proyecto de tesis para realizar el diseño del flujo de ejecución de los algoritmos.

1.3.2 Netbeans

Es un entorno de desarrollo integrado libre y de código abierto proporcionado por Apache que puede ser instalado en cualquier sistema operativo que soporta Java; es decir, Windows, Linux o Mac OS X. Facilita el desarrollo de aplicaciones de escritorio, móviles y web, utilizando

diversos lenguajes de programación tales como Java, JavaScript, HTML5, PHP y C/C++. (NetBeans, s.f)

Se utilizará este entorno de trabajo en el proyecto de tesis para la implementación de los algoritmos genético y Cuckoo Search, así como para la interfaz gráfica.

1.3.3 Java

Java es un lenguaje de programación derivado del lenguaje C y una plataforma informática que muchas aplicaciones y sitios web utilizan para funcionar. Se puede encontrar en computadoras portátiles, centros de datos, consolas para juegos e incluso súper computadoras, ya que es rápido, seguro y fiable (Oracle, s.f)

Se utilizará para la implementación de los algoritmos y la interfaz gráfica del proyecto de tesis.

1.3.4 Git

Git es un sistema de control de versiones libre y de código abierto, diseñado para manejar desde pequeños hasta muy extensos proyectos de desarrollo de software con eficiencia. Sus principales características son la ramificación y la facilidad de trabajar múltiples flujos en paralelo (Git, s.f)

Se utilizará en el proyecto de tesis como repositorio de versiones del código fuente de los algoritmos y la interfaz gráfica.

1.3.5 Scrum

Scrum es un marco de trabajo iterativo e incremental para el desarrollo de proyectos, cuya estructura se desarrolla en ciclos de trabajo llamados Sprints, los cuales son iteraciones de 1 a 4 semanas que se van sucediendo una detrás de otra. Al comienzo de cada Sprint, un equipo multi-funcional selecciona elementos de una lista de tareas, llamada Sprint Backlog, y los prioriza para ser alcanzados en el Sprint más cercano. La comunicación, bajo esta metodología,

es constante con las partes interesadas, ya que se pretende desarrollar entregables incrementales que generen valor para el usuario. (Deemer et al., 2009).

Para el desarrollo del proyecto de tesis se utilizarán algunas buenas prácticas que esta metodología sugiere, tales como la planificación de iteraciones cortas y la comunicación constante con el asesor, con la finalidad de recibir retroalimentación suficiente para mantener el enfoque deseado de la solución propuesta.

1.3.6 ANOVA

El análisis de varianza (ANOVA) es un método paramétrico para el análisis de datos de experimentos. Está diseñado para evaluar las diferencias de las medias entre diferentes grupos o tratamientos, evitando, de esta manera, tener que realizar múltiples comparaciones por cada grupo. Tiene tres requisitos que deben cumplirse en el conjunto de datos para poder aplicarlo: las muestras deben ser independientes, la varianza debe ser igual para los grupos y, por último, estos deben seguir una distribución normal (Armstrong, Eperjesi & Gilmartin, 2002).

Se utilizará en el proyecto de tesis para realizar la experimentación numérica y realizar la comparación de los resultados de los algoritmos.

1.3.7 Pruebas estadísticas no paramétricas

Las pruebas no paramétricas se utilizan para analizar un conjunto de datos de experimentos cuando los requisitos para realizar una prueba paramétrica no se cumplen. Estos métodos proveen diversas alternativas estadísticas que no tienen, o tienes muy pocos, requisitos sobre la naturaleza de los datos a evaluar. Algunas de las más conocidas son la prueba de Wilcoxom, Kruskal Wallis y Friedman (Whitley & Ball, 2002).

1.3.8 RStudio

RStudio es un entorno de desarrollo integrado (IDE) libre y de código abierto para el lenguaje de programación R. Provee diversas herramientas para el tratamiento de datos, cálculos y gráficos estadísticos (RStudio, s.f).

Se utilizará en el proyecto de tesis para realizar diversos análisis estadísticos sobre los resultados de los algoritmos desarrollados.

Capítulo 2. Marco Legal/Regulatorio/Conceptual/otros

2.1 Objetivo

El marco conceptual presentado a continuación tiene por objetivo describir los principales conceptos relacionados al problema de planificación de citas de recepción de materia prima para una empresa productora multiplanta. Las definiciones mostradas serán contextualizadas mediante ejemplos para facilitar su compresión y vinculación con el mismo.

2.2 Desarrollo del marco

2.2.1 Almacén

Un almacén es un espacio dedicado al almacenamiento y manejo de bienes y materiales, donde el orden y rendimiento del mismo es más importante que el propio almacenamiento físico, especialmente en operaciones de rotación constante de inventario. Por lo tanto, el énfasis reside en el planeamiento de las actividades del almacén, incluyendo recepción, almacenamiento, ensamblaje, recojo y despacho. De esta manera el almacén se convertirá en un importante factor de valor agregado a los servicios brindados (Emmett, 2005).

Figura 1. Representación de "Limpieza industrial de almacenes: ¿cómo mantener el orden?". Adaptado de (Mecalux, 2019).

Existen almacenes de diversos tipos; por ejemplo, de materia prima, productos intermedios o terminados. Cuando estos bienes son requeridos, se extraen del almacén y se utilizan en el proceso productivo correspondiente.

2.2.2 Inventario

Los inventarios constituyen un recurso en términos de bienes almacenados del cual se valen las organizaciones para satisfacer una demanda en el futuro. Los motivos básicos para crear inventarios son protegerse contra incertidumbres, permitir la producción y compra bajo condiciones económicamente ventajosas, cubrir cambios anticipados en la demanda y la oferta y mantener el tránsito entre los puntos de producción o almacenamiento (Gutiérrez, 2009).

La gestión de inventarios se deriva de la importancia que tienen las existencias para la empresa y, por lo tanto, la necesidad de administrarlas y controlarlas considerando la capacidad de almacenamiento disponible. Su objetivo consiste fundamentalmente en mantener un nivel de inventario que permita, a un mínimo de costo, un máximo de servicio a los clientes (Gutiérrez, 2009).

Se pueden mantener inventarios de diversos bienes; por ejemplo, inventario de materia prima, aditivos, preservantes e insumos químicos para la producción.

2.2.3 Materia prima

La materia prima se define como elementos extraídos de la naturaleza que, en el sector industrial, serán utilizados como insumos para la fabricación de bienes o productos mediante un proceso de transformación (Emmett, 2005). La lana, el algodón o la madera son ejemplos de materia prima utilizada en procesos industriales.

2.2.4 Productora multiplanta

Las empresas multiplanta son aquellas en las que las actividades productivas se distribuyen en una red de plantas en diferentes localidades, las cuales se complementan o distribuyen la producción, con la finalidad de que exista una colaboración entre las mismas (Alvarez, 2007). Este factor añade una importante complejidad al realizar la planificación, tanto de producción como de abastecimiento, ya que, al compartir los recursos y necesidades, mantener la información actualizada es vital para la planificación de tareas; de lo contrario, no será posible optimizar al sistema en conjunto (Alvarez, 2007).

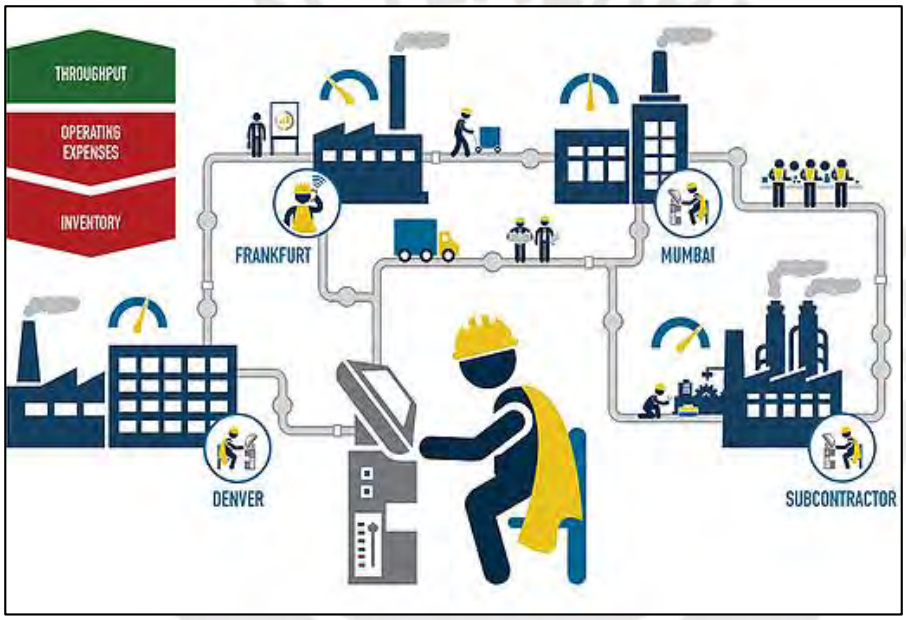


Figura 2. Representación de "Programación de múltiples plantas para generar ganancias". Adaptado de (Planet Together, s.f).

Es muy común en la actualidad que las empresas productoras de alimentos sean multiplanta, ya que asignan la producción de determinado producto a una localidad específica. Por ejemplo, una empresa puede asignar a la planta de Lima la producción de lácteos, mientras que en Ica se producen las conservas. Estas pueden abastecerse desde un almacén central ubicado estratégicamente para reducir los costos de transporte o cada una puede tener el propio, lo cual

no significa que no puede compartir capacidad de almacenamiento en caso otra planta lo necesite.

2.2.5 Gestión de cadena de suministros

Una cadena de suministros es una red de organizaciones involucradas directamente a través de flujos operacionales en diferentes procesos y actividades que producen valor en la forma de productos o servicios a un consumidor final. Se extiende mucho más allá que simplemente el movimiento físico del material, ya que involucra gestión de abastecimiento, compras y materiales; planificación de producción, servicio al cliente y flujos tales como el proceso de distribución y transporte (Stevens, 1989).

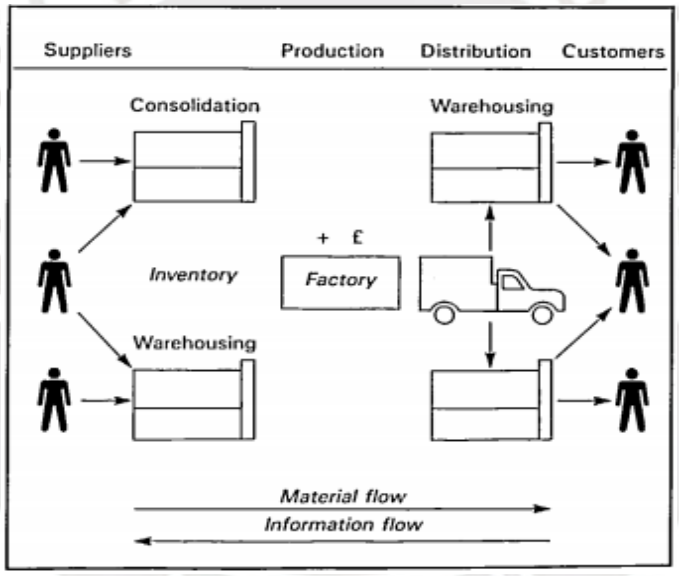


Figura 3. Representación de "El alcance de una cadena de suministros". Adaptado de (Stevens, 1989).

La gestión de la cadena de suministros es un conjunto de decisiones sincronizadas y actividades usadas para la integración eficiente de proveedores, transportistas, almacenes, productores, distribuidores y clientes finales. Esto para permitir que el producto o servicio esté disponible en la cantidad, precio, condición y tiempo correctos, con la finalidad de minimizar el costo de

todo el sistema mientras se busca satisfacer los requerimientos del cliente (Mentzer et al., 2001).

Para mostrar la integración entre actores de una cadena de suministro en el sector industrial se puede tomar como ejemplo una empresa productora de lácteos. La cual recibe desde sus proveedores; en este caso, los ganaderos, la leche fresca proveniente de las vacas. El proceso de manufactura convierte esta materia prima en un tarro de leche y finalmente se distribuye este producto terminado a los supermercados, minimarkets y bodegas. Es claro que la empresa no puede comprar leche fresca sin ningún proceso logístico de soporte que le permita mantener un nivel de inventarios adecuado según la producción planeada, ni tampoco podría satisfacer una demanda hacia los clientes sin considerar la oferta a la cual está limitada. Así pues, se requiere que el flujo operacional desde los proveedores hasta los clientes esté en constante integración para garantizar un nivel adecuado de servicio durante todo el proceso.

2.2.6 Logística

La logística, es la parte de los procesos de la cadena de suministros, que puede ser definida como la gestión estratégica de las adquisiciones, movimiento y almacenamiento de materiales, piezas e inventario terminado (y flujos de información relacionados) a través de la organización y sus canales de venta de una forma rentable. Esto a través de acciones presentes y futuras que permitan el cumplimiento de las órdenes a un costo eficiente (Anca, 2019). Mayormente es vista, estudiada e implementada como una herramienta práctica para dar soporte a los negocios y funciones de diversas organizaciones y sistemas técnicos en el alcance de los objetivos (Milenkova et al.,2020).

La logística se da a nivel interno de la organización, mientras que la gestión de la cadena de suministros abarca toda la red de organizaciones que participan en la misma. Por ejemplo, un proceso logístico aplicado al tema de estudio, es la planificación de las citas de recepción de

abastecimiento para empresas manufactureras. Esta planificación la realiza el personal encargado conociendo, de antemano, las prioridades de materia prima requerida, el stock de inventarios actual, la capacidad disponible de almacén, así como otras restricciones que se deben tomar en cuenta con la finalidad de obtener una planificación adecuada para evitar perjuicios económicos a la empresa.

2.2.7 Sistemas de Planificación de Recursos Empresariales (ERP)

Los ERP son sistemas de información que ayudan a las empresas a lograr sus objetivos a través de la integración en tiempo real. Consisten en diversos módulos, tales como ventas, recursos humanos, finanzas, producción, entre otros; las cuales están interconectadas para permitir el intercambio de información entre las diferentes unidades organizacionales (Banaeianjahromi, 2016).

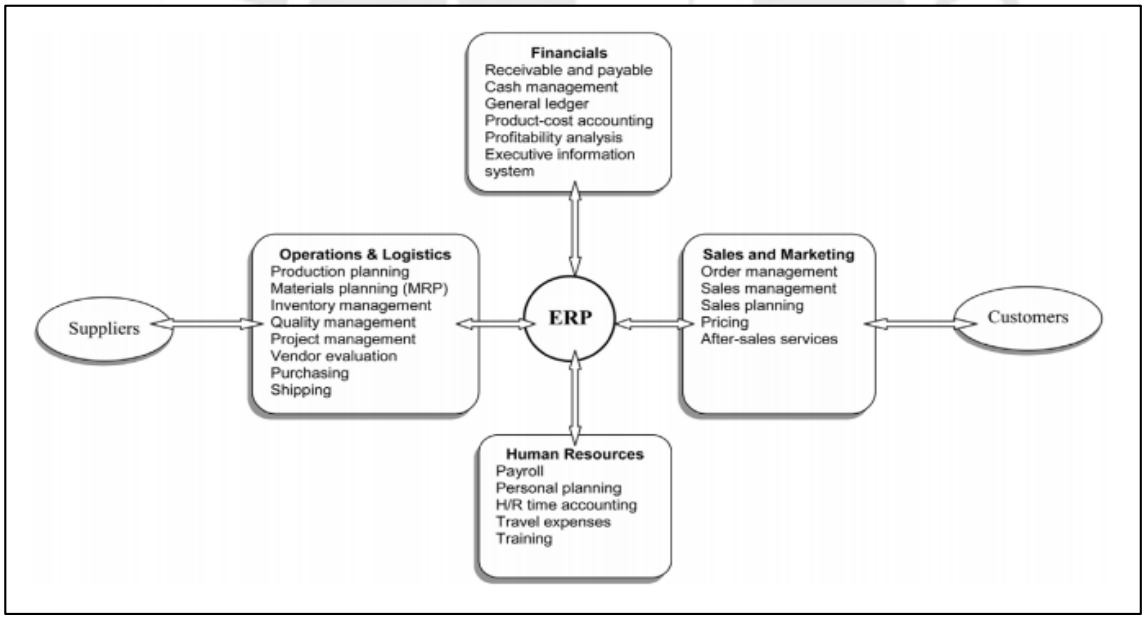


Figura 4. Representación de "Una visión general de un sistema ERP". Adaptado de (Chen, 2001).

Los sistemas de este tipo ofrecen un repositorio central de datos donde se busca reducir la redundancia, incrementar la eficiencia de la cadena de valor, el acceso del cliente a los productos y servicios, así como reducir los costos operacionales (Banaeianjahromi, 2016).

Uno de los sistemas ERP más populares a nivel mundial, por ejemplo, es SAP, el cual permite, dentro de otras funciones, recopilar los datos relacionados a aspectos operativos de la empresa y procesarlos para facilitar la toma de decisiones a los usuarios. Esta herramienta posee un módulo dedicado a logística, el cual permite controlar los procesos de abastecimiento de la empresa y gestión de niveles de inventario. Sin embargo, el uso de este sistema, en muchas ocasiones, genera retrasos y afectaciones a las actividades debido a su complejidad, la cual requiere usualmente capacitaciones constantes, con la finalidad de alinear su uso a los objetivos organizacionales (Gargeya & Brady, 2005).

2.2.8 Problemas NP-difíciles

Los problemas NP- difíciles, tan complejos como los problemas NP-completos, son aquellos en los que no se conoce un algoritmo determinístico hasta la fecha para resolverlos en un tiempo polinomial en función al tamaño del problema (Guturu & Dantu, 2008). Por lo tanto, en el peor escenario, se tendrá un incremento exponencial en el tiempo de procesamiento. Es por esto que para su resolución se usan diversas técnicas heurísticas, metaheurísticas y estrategias de optimización global, como los algoritmos genéticos, las cuales permiten obtener una solución aceptable en un tiempo razonable, con el riesgo de que esta no sea la óptima (Guturu & Dantu, 2008).

Los problemas de planificación son del tipo NP-difíciles, ya que resulta muy complejo resolverlos utilizando algoritmos tradicionales o técnicas matemáticas. Por ejemplo, en un puerto marítimo, planificar la actividad diaria involucra considerar diversos factores tales como horario de llegada de los buques, asignación de espacios para los containers en el patio y el

horario de los camiones de entrega y recojo de mercadería. Resolver este tipo de problemas con un algoritmo determinístico o un modelamiento matemático resultará muy complejo y, de poder formularlo, el tiempo de procesamiento necesario será demasiado alto e incluso es probable no encontrar una solución.

2.2.9 Problemas de optimización combinatoria

Un problema de optimización se define por:

- Un conjunto de soluciones que representa el espacio de búsqueda (D).
- Una función de evaluación f que asocia a cada solución s un valor representando su calidad.

En función del problema, la naturaleza de las soluciones $s \in D$ varía y estarán sujetas a una serie de restricciones, determinando así cuáles de ellas son factibles. En el caso de un conjunto finito de soluciones discretas, se tratará de un problema de optimización combinatoria (López, 2017).

Esta disciplina tiene numerosas aplicaciones a nivel de industrias, social, administración de organizaciones, entre otros. Por ejemplo, la planificación de la distribución mediante un enrutamiento óptimo de vehículos es un recurrente problema del sector industrial en el cual se busca generar una solución considerando diversas restricciones como el número de vehículos, la capacidad de los mismos o las fechas de entrega límite. La combinación de estas restricciones y parámetros finalmente permitirá, mediante el uso de una herramienta eficiente, obtener un plan de distribución adecuado para el modelo.

2.2.10 Problemas de planificación

Los problemas de planificación son una sub-clase de problemas de optimización combinatoria en campos tales como operaciones de producción y despacho en la industria manufacturera y extractiva, donde el objetivo principal de los investigadores es la reducción de los costos de

producción en la industria (Pandolfi et al., 2018). Este tipo de problemas incluyen la combinación de recursos, tareas, objetivos y restricciones. Muchos de los problemas de planificación son computacionalmente complejos y el tiempo requerido para calcular una solución óptima se incrementa con el tamaño del problema. Además, se ha demostrado que este tipo de problemas pertenecen a la clase NP-difícil (Pandolfi et al., 2018).

Existen numerosas aplicaciones de problemas de este tipo, entre los más populares destacan, por ejemplo, la planificación de la producción. Este problema consiste en encontrar la secuencia óptima de las tareas de producción con la finalidad de asignar las máquinas y recursos disponibles a los procesos buscando minimizar el tiempo total de la fabricación de un lote de determinado producto.

2.2.11 Planificación de citas

En el caso de la planificación específica de citas, se pretende conseguir, además de los objetivos de la planificación en general, reducir los tiempos de espera a los que se exponen los involucrados. Es decir, si el sistema usualmente está en un estado inactivo, no está siendo eficiente en el uso de recursos y costos, por lo que el objetivo principal de este tipo de problemas es minimizar los tiempos y las pérdidas económicas que estos representan mediante una planificación óptima de tiempos de llegada (Kemper, Klaassen & Mandjes, 2014).

Un ejemplo de este tipo es, precisamente, la aplicación que se presenta en este proyecto de fin de carrera, el cual tiene la finalidad de generar una programación adecuada para recibir la materia prima que entregan los proveedores, buscando minimizar los tiempos de espera y los sobrecostos que puede ocasionar una planificación inconsistente.

2.2.12 Algoritmo metaheurístico

Un algoritmo metaheurístico es un proceso iterativo de alto nivel el cual guía a técnicas heurísticas subordinadas mediante la combinación de conceptos tales como la exploración y

explotación de espacios de búsqueda, estructurando la información con la finalidad de obtener soluciones eficientes y cercanas al óptimo (Blum & Roli, 2003).

En los algoritmos metaheurísticos es muy importante mantener un balance entre la capacidad de exploración y el aprovechamiento de la experiencia acumulada, ya que, el primero permite identificar regiones del espacio de búsqueda con soluciones de alta calidad y el segundo evita desperdiciar tiempo en regiones que ya han sido exploradas o que no representan una mejora a la solución (Blum & Roli, 2003).

Las estrategias de búsqueda de los diferentes algoritmos metaheurísticos dependen de la filosofía del mismo; por ejemplo, existen algoritmos bio-inspirados tales como el algoritmo genético o la colonia de hormigas, los cuales basan su idea central en la imitación del comportamiento de elementos naturales.

2.2.13 Algoritmo genético

El algoritmo genético, desarrollado por John Holland y colaboradores en 1960, es un modelo de abstracción de evolución biológica basado en la teoría de la selección natural de Charles Darwin. Utiliza operadores genéticos tales como la recombinación, mutación y selección, los cuales son parte esencial del algoritmo (Yang, 2010).

A partir de esta idea, se han desarrollado numerosas variantes de este algoritmo y ha sido aplicado a diferentes tipos de problemas de optimización relacionados a diversas áreas de estudio, ya que ha demostrado tener la capacidad de adaptarse a diferentes contextos y objetivos de forma satisfactoria (Yang, 2010).

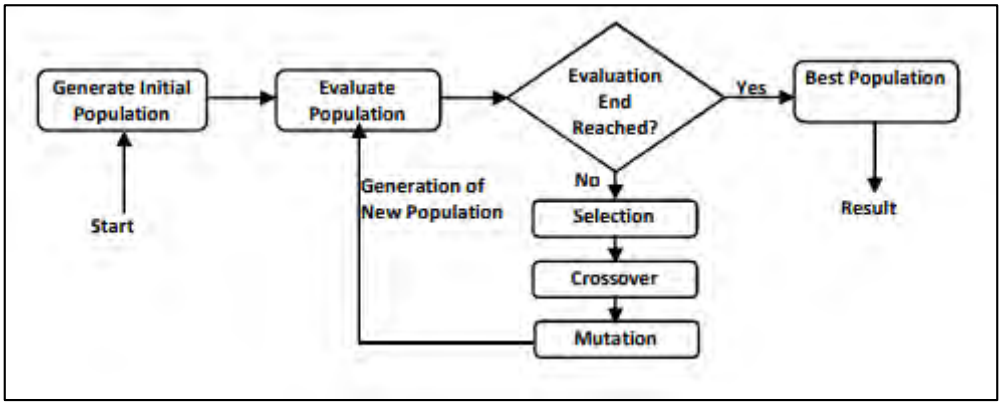


Figura 5. Representación de "Estructura y procesamiento de un algoritmo genético". Adaptado de (Majeed & Kumar, 2014).

2.2.14 Algoritmo Cuckoo Search

El algoritmo Cuckoo Search es uno de los más reciente algoritmos metaheurísticos bio-inspirados y fue desarrollado en el 2009 por Xin-She Yang y Suash Deb. Está basado en la estrategia de reproducción de algunas especies de Cuckoos, los cuales depositan sus huevos en nidos pertenecientes a otras aves, con el objetivo de que estos puedan aprovechar los recursos que el anfitrión le brinde como si fuesen propios, permitiendo que estos maduren y puedan repetir el procedimiento con sus crías (Yang, 2010).

Diversos estudios que aplicaron este algoritmo en la resolución de problemas de optimización demuestran de que es potencialmente superior y más eficiente que algoritmos populares como el enjambre de partículas y el algoritmo genético (Yang, 2010).

Objective function $f(\boldsymbol{x})$, $\boldsymbol{x} = (x_1, ..., x_d)^T$
Generate initial population of n *host nests* $\boldsymbol{x}_i$
while *(*$t <$ *MaxGeneration) or (stop criterion)*
  Get a cuckoo randomly/generate a solution by Lévy flights
    and then evaluate its quality/fitness F_i
  Choose a nest among n *(say,* j*) randomly*
  if $(F_i > F_j)$,
    Replace j *by the new solution*
  end
  A fraction (p_a) *of worse nests are abandoned*
    and new ones/solutions are built/generated
  Keep best solutions (or nests with quality solutions)
  Rank the solutions and find the current best
end while
Postprocess results and visualization

Figura 6. Representación de "Pseudocódigo de Cuckoo Search". Adaptado de (Yang, 2010).

Capítulo 3. Estado del Arte

3.1 Introducción

El Estado del Arte presentado a continuación permitirá conocer más a detalle aplicaciones relacionadas al tema de investigación seleccionado. Se presentarán las preguntas de investigación formuladas, así como las bases de datos de las que se extraerá la información. Finalmente, mediante un formulario diseñado para filtrar los documentos relevantes se procederá a dar respuesta a las preguntas planteadas.

3.2 Objetivos de revisión

La siguiente revisión se ha elaborado con el objetivo de conocer cuáles son las herramientas que se utilizan para resolver problemas de planificación referentes a operaciones logísticas en el sector industrial y optimizar la misma utilizando el menor recurso computacional. Se pretende obtener un conocimiento reciente de las propuestas trabajadas en los últimos años, en este campo de estudio, lo que permitirá garantizar que la solución desarrollada en este proyecto de fin de carrera representa un aporte de valor significativo y responde a una necesidad importante en el sector estudiado.

Para realizar la revisión sistemática del estado del arte se ha seguido la metodología elaborada por Kitchenham (Kitchenham, 2004), con la finalidad de realizar una investigación objetiva e identificar documentos relevantes para el estudio a través de criterios definidos.

3.3 Preguntas de revisión

Para estructurar la formulación de las preguntas de revisión y la búsqueda de información, se utiliza la estrategia PICOC que permite definir los principales conceptos de la investigación y enfocar adecuadamente el propósito de la misma.

Tabla 12. Criterios PICOC.

Criterios PICOC	
Criterio	**Descripción**
Población	Aplicaciones informáticas utilizadas para la planificación de operaciones logísticas.
Intervención	Algoritmos metaheurísticos utilizados para la resolución de problemas de planificación de citas en operaciones logísticas del sector industrial.
Comparación	Algoritmo Cuckoo Search respecto a soluciones informáticas más frecuentes al problema.
Resultados	Casos de estudio donde se hayan aplicado soluciones exitosas con un bajo consumo computacional para resolver problemas de planificación de citas en operaciones logísticas del sector industrial.
Contexto	Académico y empresarial.

Fuente: Elaboración propia.

Con base en lo expuesto en el objetivo de la revisión y los criterios establecidos por la estrategia PICOC, se formularon las siguientes preguntas de investigación:

- P1. ¿Cuáles son las aplicaciones recientes que se utilizan para resolver un problema de planificación relacionado a operaciones logísticas en empresas industriales?
- P2. ¿Qué aplicaciones informáticas basadas en metaheurísticas se usan para resolver un problema de planificación de citas relacionado a operaciones logísticas en empresas industriales?
- P3. ¿Cuáles son las aplicaciones del algoritmo Cuckoo Search orientadas resolver un problema de planificación de citas relacionado a operaciones logísticas en empresas industriales?

3.4 Estrategia de búsqueda

3.4.1 Motores de búsqueda a usar

Se seleccionaron las siguientes bases de datos para realizar la revisión sistemática ya que se consideran las más relevantes para la búsqueda de información en el área de Ingeniería Informática y por recomendación del asesor con experiencia en investigación.

- Scopus
- IEEE Explore
- ACM Digital Library

3.4.2 Cadenas de búsqueda a usar

Para elaborar la cadena de búsqueda se descompusieron las preguntas de investigación en sus criterios PICOC y se realizaron búsquedas preliminares utilizando varias combinaciones de términos derivados, así como la revisión de títulos y resúmenes de los estudios recuperados. Se utilizan palabras en inglés ya que las bases de datos utilizadas tienen, principalmente, documentos en este idioma.

Tabla 13. Términos de búsqueda por criterio.

Términos de búsqueda por criterio	
Criterio	**Términos**
Herramientas	Software / System / Systems / Tool / Tools / Solution
Metaheurísticas	Metaheuristic Algorithm / Cuckoo Search / Cuckoo-Search / Cuckoo
Planificación	Scheduling / Schedule / Planning
Cadena de suministro	Supply Chain / Logistic / Logistics / Industrial

Fuente: Elaboración propia.

Generando la siguiente cadena resultante, la cual se adaptará a la sintaxis de cada una de las bases de datos a utilizar.

("software" OR "system" OR "tool*" OR "solution") AND ("metaheuristic algorithm" OR "cuckoo search" OR "cuckoo-search" OR "cuckoo") AND ("schedul*" OR "planning") AND ("supply chain" OR "logistic*" OR "industrial")*

SCOPUS: TITLE-ABS ((software OR system* OR tool* OR solution) AND ("metaheuristic algorithm" OR "cuckoo search" OR "cuckoo-search" OR cuckoo) AND (schedul* OR planning) AND ("supply chain" OR logistic* OR industrial))

IEEE EXPLORE: (software OR system* OR tool* OR solution) AND ("metaheuristic algorithm" OR "cuckoo search" OR "cuckoo-search" OR cuckoo) AND (schedul* OR planning) AND ("supply chain" OR logistic* OR industrial)

ACM DIGITAL LIBRARY: (software OR system* OR tool* OR solution) AND ("metaheuristic algorithm" OR "cuckoo search" OR "cuckoo-search" OR cuckoo) AND (schedul* OR planning) AND ("supply chain" OR logistic* OR industrial)

3.4.3 Documentos encontrados

Luego de ejecutar la búsqueda se obtuvieron los siguientes resultados en cada una de las bases de datos:

Tabla 14. Resultados de ejecución de búsqueda.

Resultados de ejecución de búsqueda			
Base de datos	**Documentos encontrados**	**Documentos repetidos**	**Documentos relevantes**
Scopus	53	0	8
IEEE Explore	15	2	1
ACM Digital Library	83	0	1
Total	151	2	10

Fuente: Elaboración propia.

3.4.4 Criterios de inclusión/exclusión

Para considerar relevante un documento se definieron los siguientes criterios de inclusión luego de analizar los resultados obtenidos en la búsqueda:

1. El artículo presenta un caso de estudio de un problema de planificación del sector industrial o empresarial.
2. El artículo presenta el empleo del algoritmo Cuckoo Search u otras herramientas para resolver el caso de estudio.
3. El artículo está redactado en inglés o español.

Asimismo, se definieron los siguientes criterios de exclusión:

1. El artículo tiene una antigüedad mayor a 5 años, ya que se prefiere utilizar documentos recientes.
2. Artículos no relacionados a soluciones informáticas.
3. El artículo es redundante. Es decir, ya se ha seleccionado, o excluido, un caso de estudio igual o muy similar.

3.5 Formulario de extracción de datos

El siguiente formulario se diseñó para facilitar el análisis y síntesis de la información con la finalidad de filtrar adecuadamente los documentos relevantes que permitirán responder a las preguntas planteadas.

Tabla 15. Formulario de extracción de datos.

Campo	Descripción	Pregunta de revisión
Id	Identificador. Ej. A01	General

Base de datos	Nombre de la base de datos	General
Autores		General
Título		General
Año de publicación		General
Problemas abordados	Qué problema relacionado a la logística o al sector industrial se está tratando de resolver.	General
Contextualiza escenario	Presenta escenarios similares o contextualiza el caso de estudio con otros relacionados.	General
Proceso logístico	Identifica el proceso logístico abordado y los conceptos relacionados.	General
Herramientas utilizadas	Nombre de los sistemas, tecnologías u otras herramientas utilizadas.	P1
Integra soluciones	Integra la solución planteada con otra técnica, algoritmo, metodología.	P1
Metaheurísticas utilizadas	Nombre de las metaheurísticas utilizadas.	P2
Aplica Cuckoo Search	Aplica el algoritmo Cuckoo Search como parte de la solución propuesta.	P3
Adaptación al problema	Adapta el problema al algoritmo en forma matemática, formula los parámetros y funciones de evaluación.	P2, P3
Análisis de resultados	Ejecuta el algoritmo y analiza resultados obtenidos.	P2, P3
Compara algoritmos	Compara el algoritmo Cuckoo Search respecto a otras soluciones informáticas.	P3

Fuente: Elaboración propia.

3.6 Resultados de la revisión

Luego de analizar los resultados obtenidos se seleccionaron los documentos considerados relevantes para el estudio, los cuales se detallan en la Tabla 16.

Tabla 16. Documentos relevantes para el estudio.

Id	Base de datos	Autores	Año de publicación	Título
A01	Scopus	Alvarez, A., Munari, P., & Morabito, R.	2018	Iterated local search and simulated annealing algorithms for the inventory routing problem.
A02	Scopus	Cohen, M. W., Coelho, V. N., Dahan, A., & Kaspi, I.	2017	Container vessel stowage planning system using genetic algorithm.
A03	Scopus	Dulebenets, M. A.	2019	A Delayed Start Parallel Evolutionary Algorithm for just-in-time truck scheduling at a cross-docking facility.
A04	Scopus	Jamili, N., Ranjbar, M., & Salari, M.	2016	A bi-objective model for integrated scheduling of production and distribution in a supply chain with order release date restrictions.
A05	Scopus	Kisialiou, Y., Gribkovskaia, I., & Laporte, G.	2019	Supply vessel routing and scheduling under uncertain demand.
A06	Scopus	Maadi, M., Javidnia, M., & Ramezani, R.	2018	Modified Cuckoo Optimization Algorithm (MCOA) to solve Precedence Constrained Sequencing Problem (PCSP).
A07	Scopus	Sangaiah, A. K., Tirkolaee, E. B., Goli, A., & Dehnavi-Arani, S.	2019	Robust optimization and mixed-integer linear programming model for LNG supply chain planning problem.

A08	Scopus	Mar-Ortiz, J., Castillo-García, N., & Gracia, M. D.	2020	A decision support system for a capacity management problem at a container terminal.
A09	IEEE Explore	Sallam, K. M., Chakrabortty, R. K., & Ryan, M. J	2019	A Hybrid Differential Evolution with Cuckoo Search for Solving Resource Constrained Project Scheduling Problems.
A10	ACM Digital Library	Amirghasemi, M., & Zamani, R.	2017	An Effective Evolutionary Hybrid for Solving the Permutation Flowshop Scheduling Problem

Fuente: Elaboración propia.

3.6.1 Respuesta a pregunta P1

Para dar respuesta a la primera pregunta de investigación "¿Cuáles son las aplicaciones recientes que se utilizan para resolver un problema de planificación relacionado a operaciones logísticas en empresas industriales?" se seleccionaron los documentos A01, A02, A03, A04, A05, A07, A08 y A10. En el Anexo B, "Formulario de extracción de datos", se puede revisar el contenido de los mismos.

3.6.1.1 Análisis de resultados

A través de la revisión sistemática se pudieron identificar las aplicaciones recientes que se utilizan en el sector industrial para resolver problemas asociados a decisiones logísticas. La aplicación de algoritmos metaheurísticos resultó ser la más frecuente para este tipo de problemas, ya que, dado que se requiere manejar un amplio conjunto de restricciones y debido a su capacidad de ser altamente parametrizables, permiten diseñar el problema de manera eficiente y obtener soluciones de gran calidad (Sangaiah et al., 2019). Por ejemplo, en A02 se

utiliza un algoritmo metaheurístico para la asignación de espacios de containers en los buques de transporte, mientras que en A01 se utiliza para la planificación de la distribución de productos mediante un enrutamiento óptimo de vehículos. También se aplican a problemas mucho más complejos que involucran optimizar el funcionamiento de todo un sistema como en A03, donde se debe considerar diversos factores involucrados en el cross-docking, proceso logístico de distribución que no utiliza almacenes intermedios.

Se identificó mediante el análisis de los documentos relevantes que los algoritmos metaheurísticos son frecuentemente complementados con diferentes herramientas y métodos exactos, los cuales a pesar de haber mostrado claras limitaciones debido al gran número de variables y restricciones en la formulación de los problemas (Cohen, M., et al 2017), permiten obtener un análisis preliminar e incluso soluciones aceptables cuando la complejidad de los mismos es menor y los tiempos de procesamiento en la práctica no son demasiado altos (Alvarez, Munari & Morabito, 2018) . La construcción heurística en A01, la programación lineal entera en A04 y el uso de software de modelado GAMS y métodos estadísticos en A07 se suelen presentar en los casos de estudio con la finalidad de plantear una base matemática del mismo, para luego extrapolar el problema a las dimensiones reales requeridas en las empresas.

3.6.2 Respuesta a pregunta P2

Para dar respuesta a la segunda pregunta de investigación ¿Qué aplicaciones informáticas basadas en metaheurísticas se usan para resolver un problema de planificación de citas relacionado a operaciones logísticas en empresas industriales?" se seleccionaron los documentos A01, A03, A05 y A08 que se presentan en el Anexo B "Formulario de extracción de datos".

3.6.2.1 Análisis de resultados

Específicamente en los problemas que buscan resolver una planificación de citas, o bien dicha planificación es parte del problema central que se está resolviendo, se ha identificado el uso de diversos algoritmos metaheurísticos, siendo los más destacados el algoritmo genético y la búsqueda tabú. La principal característica de estos algoritmos es que evalúan un gran número de posibles soluciones debido a sus estrategias para escapar de óptimos locales, evaluando constantemente el entorno con el objetivo de aproximarse a la mejor solución.

El algoritmo genético está basado en la evolución, encargándose, en cada iteración, de realizar modificaciones a los individuos con el objetivo de obtener soluciones cada vez más prometedoras (Dulebenets, 2019). En el caso de A03, por ejemplo, ha demostrado un rendimiento satisfactorio al realizar la planificación de horarios en un ambiente real de cross-docking, técnica logística que evita el uso de almacenes intermediarios.

La búsqueda tabú, por su parte, es un procedimiento de búsqueda adaptable y diseñado para, a partir de una solución inicial, recorrer todas las soluciones vecinas con la finalidad de optimizar la calidad de la misma (Mar-Ortiz, Castillo-García & Gracia, 2020). En A08, se utiliza para realizar la planificación del tráfico de camiones en un terminal marítimo, de tal manera de que se reduzca el tiempo de espera de los vehículos y la administración de los recursos del puerto sea eficiente.

3.6.3 Respuesta a pregunta P3

Para dar respuesta a la tercera pregunta de investigación ¿Cuáles son las aplicaciones del algoritmo Cuckoo Search orientadas resolver un problema de planificación de citas relacionado a operaciones logísticas en empresas industriales?" se seleccionaron los documentos A06, A07 y A09 que se presentan en el Anexo B "Formulario de extracción de datos".

3.6.3.1 Análisis de resultados

Este algoritmo bio-inspirado está basado en la estrategia de reproducción de los Cuckoos, la cual consiste en la selección de nidos huéspedes donde estos dejarán sus huevos, los cuales representan las posibles soluciones, para su incubación, de tal forma que el ave anfitriona los críe como si fuesen suyos (Maadi, Javidnia & Ramezani, 2018). Con el transcurso del tiempo, las soluciones convergerán hacia los mejores nidos y de esta manera el algoritmo enfoca su búsqueda en encontrar la de mejor calidad.

Se ha identificado mediante la revisión sistemática que el algoritmo metaheurístico Cuckoo Search se utiliza en diversos problemas de planificación, ya que, a pesar de su reciente desarrollo, ha demostrado ser un potente algoritmo evolutivo capaz obtener excelentes resultados en este tipo de problemas (Maadi, Javidnia & Ramezani, 2018). Sin embargo, para fines de la revisión, en los casos de estudio donde se identificó una aplicación del algoritmo Cuckoo Search no se plantea un problema de planificación de citas ni relacionados. Principalmente se busca solucionar problemas de planificación de manufactura, asignación de tareas o enrutamiento óptimo de transporte.

3.7 Conclusiones

Luego de realizar la revisión y el análisis del Estado del Arte, se puede afirmar que el enfoque del proyecto de fin de carrera a desarrollar resultará relevante para el sector estudiado, ya que, siendo los algoritmos metaheurísticos la principal herramienta para resolver problemas de planificación debido a su alta complejidad, todavía no se ha realizado una aplicación del recientemente desarrollado algoritmo Cuckoo Search para solucionar una planificación de citas en el contexto industrial. De esta manera, y con la solución propuesta en este proyecto, se tiene por finalidad cubrir el vacío identificado.

Capítulo 4. Parámetros, restricciones y función objetivo

4.1 Introducción

En este capítulo se desarrolla el objetivo 1, el cual está compuesto por los resultados esperados R1.1 y R1.2. El primero consiste en la definición de los parámetros y restricciones para el problema de planificación. En base a estos, para el segundo resultado se elabora la formulación de la función objetivo, la cual permite la evaluación de posibles soluciones.

4.2 Resultados alcanzados

En este apartado se presenta el detalle de lo realizado para alcanzar los resultados esperados R1.1: "Lista de definición de parámetros y restricciones para el problema de planificación", y R1.2: "Formulación de la función objetivo para el problema de planificación".

4.2.1 Lista de definición de parámetros y restricciones para el problema de planificación

El contenido de esta sección está dividido en dos partes relacionadas al mismo resultado esperado (R1.1). Los parámetros del problema, en primer lugar, y, a continuación, las restricciones del mismo, que permitirán validar si una planificación es o no adecuada para ser aceptada como posible solución.

4.2.1.1 Parámetros del problema

En base al análisis realizado al problema que se desea resolver y a las variables identificadas como relevantes para el mismo, se ha definido el siguiente conjunto de parámetros que se presenta en la Tabla 17.

Tabla 17. Parámetros para el problema de planificación.

Nombre	Descripción
N	Número de días de planificación.
H	Número de horas de trabajo por día.

A	Número de almacenes en planta.
M	Número de recepciones de materia prima.
Ur	Número de unidades recibidas de materia prima (ej. 20 bolsas)
Us	Número de unidades totales solicitadas de materia prima (ej. 100 bolsas)
$CAlm$	Capacidad de almacén disponible.
FC	Fecha de compra de materia prima.
FI	Fecha de ingreso de importación de materia prima.
FR	Fecha de recepción de materia prima.
LN	Fecha límite de necesidad de materia prima.
$CVeh$	Número de unidades vehiculares recibidas en un día.
$MaxCVeh$	Número máximo de unidades vehiculares permitidas en un día.
$TpUr$	Tipo de unidad de materia prima recibida.
$TpAlm$	Tipo de unidad de almacén.
d	Número de días de margen de seguridad.
man	Recepción "m" del almacén "a" en el día "n".

Fuente: Elaboración propia.

4.2.1.2 Restricciones del problema

De igual manera, las restricciones se han definido luego de realizar un análisis del contexto estudiado, identificando para el problema, las condiciones que las posibles soluciones deben cumplir, así como las que se desean evitar, con la finalidad de que resulten útiles para los objetivos planteados.

1. La sumatoria de unidades recepcionadas en un día para cada tipo de materia prima no debe superar la capacidad disponible del almacén que las recibe.

$$\sum_{m=1}^{M} Ur_{man} \leq CAlm_{an}, \forall\, a \in A, \forall\, n \in N$$

2. La fecha de recepción de una materia prima debe estar entre la fecha de compra, o fecha de ingreso de importación, y la fecha límite de necesidad, más un margen de seguridad de "d" días.

$$(FC\ o\ FI)_m \leq FR_m + d \leq LN_m\ , \forall\ m \in M$$

3. Si una materia prima tiene fecha de ingreso de importación; es decir, proviene de otro país, la programación de su recepción debe ser el mismo día.

$$FR_m = FI_m, \forall\ m \in M\ donde\ FI_m > 0$$

4. Solo se recibe una unidad vehicular por cada hora de trabajo.

$$CVeh_{anh} = 1, \forall\ a \in A, \forall\ n \in N, \forall\ h \in H$$

5. El tipo de materia prima recepcionada debe coincidir con el tipo de almacén al cual se asigna su recepción.

$$TpUr_m = TpAlm_m, \forall\ m \in M$$

6. Se debe recepcionar el total de cada materia prima antes de la fecha límite de necesidad de la misma.

$$\sum_{n=FR}^{LN_m} Ur_{man} = Us_m, \forall\, m \in M, \forall\, a \in A$$

4.2.2 Formulación de la función objetivo para el problema de planificación

El objetivo principal del problema de planificación es minimizar el rango de tiempo en el que se recibe la totalidad de una necesidad. Es decir, para cada compra de materia prima, lo que se pretende es recibir la mayor cantidad posible de esta en días cercanos a la fecha límite de necesidad, manteniendo un margen de días de seguridad establecido. De esta manera, el tiempo de almacenamiento para cada materia prima será menor y se utiliza el recurso de la capacidad disponible del almacén asignado con mayor eficiencia.

Se ha formulado la siguiente función objetivo para formalizar lo expresado:

$$min \sum_{n=1}^{N} \sum_{a=1}^{A} \frac{\sum_{m=0}^{M} [Ur * (LN - FR)]_{man}}{CAlm_{an}}$$

4.2.3 Validaciones realizadas

Se ha realizado la evaluación de diversas soluciones para el problema utilizando la función objetivo presentada y validando que los resultados obtenidos son acordes a lo esperado. Es decir, cuando una solución presenta una mejora respecto a otra, la función objetivo se optimiza, de lo contrario, se ve afectada. En el anexo C se presenta mayor detalle sobre las pruebas realizadas.

Este capítulo ha sido revisado por el especialista en logística, quien ha dado su aprobación respecto a la información presentada en el mismo. En el anexo D se adjunta la conformidad.

4.3 Discusión

En este capítulo se han alcanzado los resultados esperados R1.1 y R1.2, el primero de ellos en los ítems 4.2.1 y 4.2.2, mientras que el segundo se describe en el apartado 4.2.3.

El alcance de estos resultados permite formalizar un planteamiento inicial de parámetros relevantes y condiciones que permitirán diseñar las estructuras de datos y los algoritmos en las siguientes etapas. Además, tener la función objetivo definida es muy importante, ya que esta es la encargada de evaluar las posibles soluciones al problema.

Se mantiene la consistencia de resultados respecto a investigaciones anteriores y casos de estudio seleccionados en el Estado del Arte, los cuales describen la aplicación de uno o más algoritmos metaheurísticos a problemas de complejidad similar, ya que estos también inician con la definición de parámetros, restricciones y la formulación de la función objetivo para el problema estudiado.

Los resultados presentados se han adecuado para resolver el problema de planificación descrito; sin embargo, realizando los ajustes necesarios, es posible generalizarlo para definir las condiciones iniciales de trabajo para problemas, por ejemplo, de optimización de capacidad disponible en almacenes o una planificación de citas para un contexto diferente, ya que estos escenarios comparten ciertas similitudes en su definición.

Finalmente, se debe indicar que tanto los parámetros, restricciones y la función objetivo tienen un enfoque individual de solución. Es decir, pese a procesar un conjunto de datos que involucra a toda la red de plantas, el planteamiento diseñado se ha definido para ser ejecutado independientemente y obtener una planificación para cada una. La otra forma de trabajar este problema, considerando un único centro de distribución, el cual reparte las recepciones a cada planta solicitante, no es parte del alcance de este proyecto.

Capítulo 5. Estructuras de datos

5.1 Introducción

En este capítulo se desarrollan los resultados esperados R2.1 y R3.1, relacionados al diseño de las estructuras de datos que dan soporte a la implementación de los algoritmos genético y Cuckoo Search según corresponde. Se presentan también las estructuras auxiliares que se utilizan para almacenar y actualizar los datos del problema.

5.2 Resultados alcanzados

En este apartado se presenta el detalle de lo realizado para alcanzar los resultados esperados R2.1: "Diseño de estructuras de datos que sirven de soporte para el algoritmo genético adaptado al problema de planificación", y R3.1: "Diseño de estructuras de datos que sirven de soporte para el algoritmo Cuckoo Search adaptado al problema de planificación".

5.2.1 Diseño de estructuras de datos que sirven de soporte para el algoritmo genético adaptado al problema de planificación

Para el algoritmo genético, el cromosoma es la representación de una posible solución al problema y, dado que en este proyecto se tiene como finalidad realizar la planificación de citas de recepción de materia prima, la estructura del mismo debe considerar los espacios de tiempo a los cuales es posible asignar una recepción, de tal forma que esta tarea pueda ser optimizada durante la ejecución del algoritmo.

Considerando estos aspectos se ha definido la siguiente estructura:

Día 1					**Día 2**					**…**	**Día n**				
V1	-	V2	…	V5	V9	-	V10	…	-	…	V4	V6	V11	…	V14

Figura 7. Representación del cromosoma para el algoritmo genético. Fuente: Elaboración propia.

Se utilizará un arreglo unidimensional como el presentado en la Figura 7, en el cual, para cada día de planificación, las celdas dentro del mismo representan las horas de trabajo. Así, la primera hace referencia al horario entre las 08:00 – 09:00 de la mañana y la última puede ser entre las 16:00-17:00 pm, por ejemplo, según la política de horas de trabajo en la empresa.

El valor dentro de cada una de las celdas indica el vehículo que se va a recibir en dicho espacio de tiempo. Según la figura mostrada, en el primer turno se recibe el vehículo etiquetado como "V1", el cual está asociado a una cantidad determinada de alguna materia prima. La estructura diseñada favorece a cumplir la restricción de asignar una sola unidad vehicular en cada hora de trabajo.

Es importante mencionar que las recepciones pueden ser parciales; es decir, no necesariamente se recibe la totalidad de una solicitud de materia prima en un solo vehículo, si no que existe la posibilidad de que sea en dos o más recepciones. Por lo tanto, se debe mantener un seguimiento de la cantidad recibida respecto al total de la necesidad, con la finalidad de obtener una planificación de citas que evite sobrecostos por reajustes posteriores.

5.2.2 Diseño de estructuras de datos que sirven de soporte para el algoritmo Cuckoo Search adaptado al problema de planificación

Para el algoritmo Cuckoo Search, la estructura que representa una posible solución al problema es el huevo del Cuckoo y, al respecto, existen dos aproximaciones para definir su relación con el nido al que pertenecen. La primera, en la que un conjunto de huevos conforma un nido o, la segunda, en la que existe una relación de uno a uno entre ambos. (Yang, 2010). Para este proyecto se utiliza la segunda consideración; por lo tanto, en adelante se utilizará el término "nido" para referir una posible solución al problema.

La estructura definida para el nido es la siguiente:

Día 1	Día 2	…	Día n

V1	-	V2	…	V5	V9	-	V10	…	-	…	V4	V6	V11	…	V14

Figura 8. Representación del nido para el algoritmo Cuckoo Search. Fuente: Elaboración propia.

Como se puede observar en la Figura 8, es la misma estructura que se utilizó en el apartado anterior para representar el cromosoma del algoritmo genético. Por lo tanto, se mantiene el detalle de la misma, en el cual la división principal corresponde a los días de planificación y las celdas dentro de los mismos son las horas de trabajo que los conforman. También, el valor que cada una de estas contiene, define el vehículo asignado para dicho rango de tiempo, de tal forma que, durante la ejecución del algoritmo, estos pueden ser reasignados a horarios en los que resulta más adecuada la planificación resultante.

5.2.3 Estructuras de datos auxiliares

Además de estructuras propias para cada algoritmo, como el cromosoma para el algoritmo genético y el nido para el Cuckoo Search, se utilizarán estructuras de datos auxiliares como parte de la implementación de ambos algoritmos. En el anexo E se presenta el detalle de las mismas.

5.2.4 Validaciones realizadas

Este capítulo ha sido revisado por el especialista en algoritmia, quien ha dado su aprobación respecto a la información presentada en el mismo. En el anexo F se adjunta la evidencia.

5.3 Discusión

En este capítulo se han alcanzado los resultados esperados R2.1 y R3.1, el primero de ellos en el ítem 5.2.1 y el siguiente en el 5.2.2. Además, se presentan las estructuras auxiliares que se utilizan para almacenamiento de datos y actualización de los mismos durante la ejecución de los algoritmos.

El alcance de estos resultados permite definir una representación de las soluciones para cada uno de los algoritmos trabajados en este proyecto. Este avance es muy importante, ya que el diseño realizado debe favorecer una rápida modificación de posibles soluciones, lo cual permite a los algoritmos tener un mejor rendimiento.

Respecto a investigaciones anteriores y casos de estudio seleccionados en el Estado del Arte, se ha identificado que solo algunas de estas plantean una definición concreta de cómo se va a representar una solución para el algoritmo utilizado. La mayor parte, luego de explicar las restricciones del problema y la función objetivo, continúan explicando la adaptación del algoritmo directamente.

Las estructuras presentadas se han diseñado para resolver el problema de planificación de este proyecto; sin embargo, es posible adecuar las mismas a problemas similares según las condiciones deseadas. Por ejemplo, un problema de asignación de recursos o turnos de trabajo, en los que también se desea definir el uso de determinado elemento en un espacio físico o de tiempo.

Finalmente, se debe indicar que alcance de las estructuras mostradas solo permite asignar un vehículo para cada espacio de tiempo. Así, no es posible recibir otra materia prima en un mismo turno en caso este ya haya sido asignado a otro transporte.

Capítulo 6. Adaptación del algoritmo genético

6.1 Introducción

En este capítulo se desarrolla el resultado esperado R2.2, relacionado a la adaptación del algoritmo genético al problema de planificación. Se presenta el detalle de los operadores de selección, casamiento y mutación, así como el método para realizar la depuración de la población antes de la siguiente generación.

6.2 Resultados alcanzados

En este apartado se presenta el detalle de lo realizado para alcanzar el resultado esperado R2.2: "Algoritmo genético adaptado al problema de planificación".

6.2.1 Algoritmo genético adaptado al problema de planificación

La presentación del diseño del algoritmo genético está dividida en los métodos principales que lo componen, con la finalidad de poder explicar el procedimiento de cada uno por separado.

6.2.1.1 Pseudocódigo del algoritmo genético

```
1  INICIO AlgoritmoGenetico(tamañoPoblacion, tasaCasamiento, tasaMutación, porc)
2      poblacion <- construirPoblacionInicial(tamañoPoblacion)
3      mejorActual <- obtenerMejorSolucion(poblacion)
4      porcentajeConservar <- porc
5      MIENTRAS condicionParada(tiempoMaximo,maxGeneraciones) HACER
6          casamiento(poblacion,tasaCasamiento)
7          mutacion(poblacion, tasaMutacion)
8          poblacion <- depuraPoblacion(poblacion, porcentajeConservar)
9          mejorActual <- obtenerMejorSolucion(poblacion)
10     FIN MIENTRAS
11 FIN AlgoritmoGenetico
```

Figura 9. Pseudocódigo del algoritmo genético. Fuente: Elaboración propia.

En la Figura 9 se muestra el pseudocódigo principal del algoritmo genético, el cual recibe a los tres parámetros importantes que este utiliza: el tamaño de población, la tasa de casamiento y la tasa de mutación. Se establece también un valor para el porcentaje de soluciones a conservar,

el cual es utilizado para el método de depuración de la población. De esta manera, se garantiza que se conservan las mejores soluciones hacia la siguiente generación.

Para el procedimiento presentado se genera una población aleatoria, para la cual se valida que cada solución que la conforma cumple las restricciones del problema (línea 2). Luego, mientras alguna condición de parada no se cumpla (línea 5), se realiza el casamiento (línea 6), la mutación (línea 7) y la depuración de la población (línea 8). Finalmente, se actualiza la mejor solución encontrada (línea 9) y se repite el procedimiento descrito.

6.2.1.2 Construcción de la población inicial

```
1  INICIO construirPoblacionInicial(tamañoPoblacion)
2      longCromosoma <- diasPlanificacion*horasDia;
3      cromosoma[longCromosoma] <- inicializar
4      actualizarFechaLimiteMargSeguridad(cromosoma)
5      PARA cont=1 HASTA numVehiculos HACER
6          SI esRecepcionImportada(vehiculo) ENTONCES
7              priorizarVehiculo(vehiculo)
8          SI NO
9              diaInicio <- fechaCompraRecepcion
10             diaFin <- fechaLimiteRecepcion
11             PARA i=diaFin DECRECE HASTA diaInicio HACER
12                 SI validarCapacidadAlmacen(i, unidadRecepcion, cantRecepcion) ENTONCES
13                     asignarVehiculo(vehiculo)
14                 FIN SI
15             FIN PARA
16         FIN SI
17     FIN PARA
18 FIN construirPoblacionInicial
```

Figura 10. Pseudocódigo de la construcción de un cromosoma para el algoritmo genético. Fuente: Elaboración propia.

En la Figura 10 se muestra el proceso de construcción de una posible solución al problema. Este se repite las veces necesarias hasta alcanzar el tamaño deseado de población inicial.

Primero, se inicializa la estructura del cromosoma según el horizonte de planificación deseado y las horas de trabajo diarias (líneas 2 y 3). Luego, se actualiza la fecha límite de recepción para cada materia considerando el margen de seguridad (línea 4). A continuación, se recorren todos los vehículos que deben entregar una recepción (línea 5) y, para cada uno, se verifica, en

primer lugar, si este tiene asociada una recepción importada (línea 6). En caso de que sea así, se debe priorizar al vehículo, ya que esta debe ser asignada a la fecha de ingreso de importación (7). En esta función se realizará la asignación prioritaria de la recepción a la fecha deseada, incluso reasignando otros vehículos a fechas distintas de ser necesario. Por otro lado, si no es una recepción prioritaria, se evalúa, para el rango permitido de fechas de la recepción, la asignación del vehículo a alguno de estos días (línea 13), verificando previamente la capacidad del almacén que será el encargado de recepcionar la materia prima (línea 12). En esta función de asignación, a diferencia de la que se encarga de recepciones prioritarias, se validará para cada día de la iteración, si existe algún espacio de tiempo disponible para asignar este vehículo. En el caso de que todas las horas del día estén ocupadas, se realiza el mismo procedimiento para los siguientes.

6.2.1.3 Método de selección

```
1  INICIO Seleccion(poblacion)
2      suma <- 0
3      valorAleatorio <- aleatorio(0,1)
4      PARA cont=1 HASTA tamañoPoblacion HACER
5          suma <- suma + (1-probabilidadIndividuo)/(tamañoPoblacion-1)
6          SI (suma >= valorAleatorio) ENTONCES
7              retornar poblacion[cont]
8          FIN SI
9      FIN PARA
10 FIN Seleccion
```

Figura 11. Pseudocódigo del método de selección para el algoritmo genético. Fuente: Elaboración propia.

En la Figura 11, se muestra el método de selección, el cual corresponde a la técnica de la ruleta, la cual tiene una modificación para adecuarla a una función objetivo de minimización.

Como primeros pasos, se inicializa una suma de probabilidades, la cual simula el recorrido de la ruleta (línea 2), y se genera una probabilidad aleatoria entre cero y uno (línea 3). Luego, a cada individuo de la población se le asigna una proporción de la ruleta según su valor de

adaptación (línea 5), la cual define una probabilidad determinada de ser seleccionado mediante esta técnica. La modificación comentada anteriormente se realiza en esta línea, ya que, en este caso, mientras su valor de adaptación sea menor, tendrá una mayor proporción de la ruleta y, por lo tanto, una mayor probabilidad de ser seleccionado. Finalmente, se evalúa la selección (línea 6), y se retorna este individuo si corresponde (línea 7).

6.2.1.4 Método de casamiento

```
1  INICIO Casamiento(poblacion, tasaCasamiento)
2      numCasamientos <- tamañoPoblacion*tasaCasamiento
3      pobHijos <- inicializar
4      PARA cont=1 HASTA numCasamientos HACER
5          padre1 <- seleccion(poblacion)
6          padre2 <- seleccion(poblacion)
7          hijos <- uniformCrossover(padre1,padre2)
8          PARA i=1 HASTA 2 HACER
9              SI esValido(hijos[i]) ENTONCES
10                 agregarSolucion(pobHijos, hijos[i])
11             FIN SI
12         FIN PARA
13     FIN PARA
14     juntarPoblaciones(poblacion, pobHijos)
15 FIN Casamiento
```

Figura 12. Pseudocódigo del método de casamiento para el algoritmo genético. Fuente: Elaboración propia.

En la Figura 12, se muestra el pseudocódigo del método de casamiento. El cual, en base a la tasa de casamiento definida, establece un número de casamientos a realizar en la población (línea 2). Luego, se selecciona con el método de la ruleta descrito anteriormente, dos individuos que serán los padres (líneas 5 y 6). A continuación, se aplica el casamiento mediante recombinación uniforme, generando dos hijos (línea 7). Finalmente, se valida si cada hijo cumple con las restricciones del problema (línea 9) y, de ser el caso, se agrega a la población de hijos (línea 10), la cual será incluida dentro de la población general luego de terminar la ejecución de todos los casamientos (línea 14).

Se muestra, en la Figura 13, el proceso para el casamiento de dos cromosomas, cada uno con 2 días de planificación y 8 horas de trabajo diarias. Se genera una máscara de valores aleatorios, los cuales indicarán los genes a ser intercambiados entre los padres, según una probabilidad definida previamente.

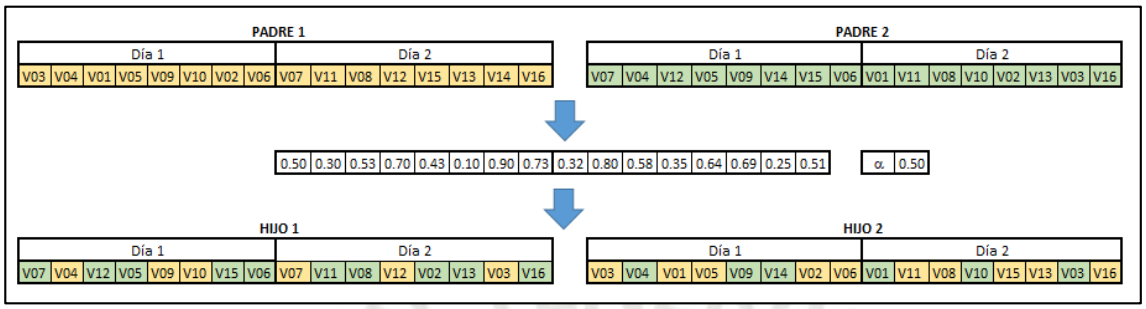


Figura 13. Ejemplo de casamiento para dos cromosomas en el algoritmo genético. Fuente: Elaboración propia.

6.2.1.5 Método de mutación

```
1  INICIO Mutacion(poblacion, tasaMutacion)
2      numMutaciones = tamañoPoblacion*tasaMutacion;
3      pobHijos <- inicializar
4      PARA cont=1 HASTA numMutaciones HACER
5          solucionarAMutar <- seleccionAleatoria(pob)
6          nuevaSolucion <- mutar(solucionAMutar)
7          SI esValido(nuevaSolucion) ENTONCES
8              agregarSolucion(pobHijos, nuevaSolucion)
9          FIN SI
10     FIN PARA
11     juntarPoblaciones(poblacion, pobHijos)
12 FIN Mutacion
```

Figura 14. Pseudocódigo del método de mutación para el algoritmo genético. Fuente: Elaboración propia.

En la Figura 14 se muestra el método de mutación, el cual define, en primer lugar, el número de mutaciones a realizar utilizando la tasa de mutación (línea 2). Luego, para el valor establecido, se realiza una iteración que se compone de los siguientes pasos: seleccionar aleatoriamente un individuo de la población (línea 5), aplicar el operador de mutación al

seleccionado y generar una nueva solución (línea 6). A continuación, se debe validar si esta solución cumple con las restricciones del problema (línea 7) y, finalmente, agregar a la población de hijos del procedimiento (línea 8).

Se muestra, en la Figura 15, el proceso de mutación para dos días de planificación. Este consiste en intercambiar posiciones al azar, validando posteriormente si cumple con las restricciones del problema.

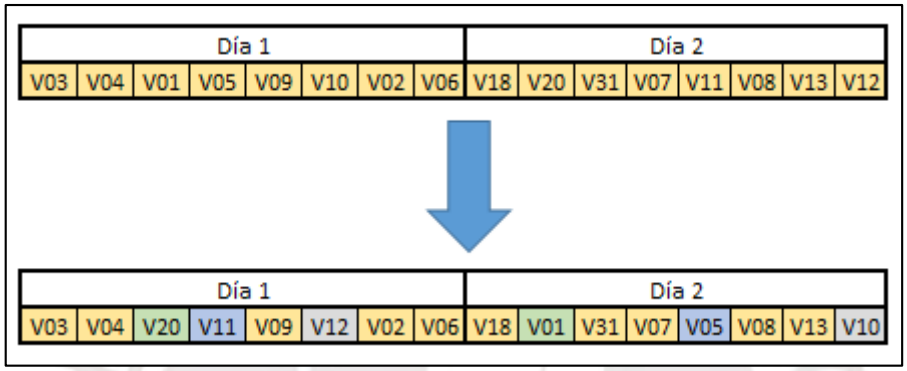


Figura 15. Ejemplo de mutación de un cromosoma para el algoritmo genético. Fuente: Elaboración propia.

6.2.1.6 Evaluación de la población

```
1  INICIO DepuraPoblacion(poblacion, porcConservacion)
2      nuevaPoblacion <- inicializar
3      cantConservar <- tamañoPoblacion*porcConservacion
4      PARA cont=1 HASTA cantConservar HACER
5          solucion <- obtenerMejorSolucion(poblacion)
6          agregarSolucion(nuevaPoblacion, solucion)
7      FIN PARA
8      PARA cont HASTA tamañoPoblacion HACER
9          solucion <- seleccion(poblacion)
10         agregarSolucion(nuevaPoblacion, solucion)
11     FIN PARA
12     retornar(nuevaPoblacion)
13 FIN depuraPoblacion
```

Figura 16. Pseudocódigo para depuración de la población en el algoritmo genético. Fuente: Elaboración propia.

En la Figura 16 se detalla el proceso de depuración de la población, el cual permite retornar a la población a su tamaño original, luego de la aplicación de los métodos presentados previamente.

En primer lugar, utilizando el porcentaje de conservación definido, se establece la cantidad de la población que se desea conservar para la siguiente generación (línea 3). Luego, para este valor, se obtienen las mejores soluciones (línea 5). A continuación, para completar la cantidad restante de la población, se selecciona mediante el método de la ruleta los individuos faltantes (línea 9). Finalmente, el conjunto de soluciones obtenidas en este procedimiento es retornada para establecer la población inicial para la siguiente generación (línea 12).

6.2.1.7 Control de aberraciones

```
1  INICIO esValido(cromosoma)
2      flagError <- 0
3      PARA i=1 hasta díasPlanificacion HACER
4          sumaAlmacenTipo <- 0
5          //Capacidad de almacenamiento
6          PARA tipoMP=1 HASTA totalTiposMP HACER
7              sumaAlmacenTipo <- sumatoriaRecepcionesTipo(cromosoma)
8              SI(sumaAlmacenTipo > capacidadMaxDiai) ENTONCES
9                  flagError <- 1
10                 SALIR
11             FIN SI
12         FIN PARA
13         //Recepciones
14         PARA recepcion=1 hasta totalRecepcionesDia HACER
15             //Fecha de importacion igual a fecha de recepcion
16             SI(esRecepcionImportada(vehiculo)) ENTONCES
17                 SI(i != fechaImportacion) ENTONCES
18                     flagError <- 1
19                     SALIR
20                 FIN SI
21             SI NO
22             //Fecha de recepcion entre fecha de compra y fecha limite
23                 SI(fechaCompraRecepcion > i) O (fechaLimiteRecepcion < i) ENTONCES
24                     flagError <- 1
25                     SALIR
26                 FIN SI
27             FIN SI
28             //Se recibe la totalidad de la solicitud en matriz de seguimiento
29             SI noCumpleTotalidad(recepcion) ENTONCES
30                 flagError <- 1
31                 SALIR
32             FIN SI
33         FIN PARA
34     FIN PARA
35 FIN esValido
```

Figura 17. Pseudocódigo del control de aberraciones para el algoritmo genético. Fuente: Elaboración propia.

En la Figura 17 se presenta el pseudocódigo para el control de aberraciones en las posibles soluciones al problema. Este procedimiento permite verificar las restricciones que se deben cumplir, con la finalidad de garantizar la calidad de los cromosomas que se utilizan durante la ejecución del algoritmo genético.

Primero, se inicializa un marcador de error (línea 2), el cual se utiliza como indicador de que se cumplen o no las restricciones al finalizar el método. Luego, se itera para cada día de planificación (línea 3) y en cada uno de estos se verifica el límite de capacidad de almacenamiento. También, para cada recepción del día que está siendo evaluado, se validan las fechas de recepción y el cumplimiento de la recepción de la totalidad de la necesidad.

Entre las líneas 6 y 12 se verifica que las recepciones del día no hayan excedido la capacidad del almacén al que cada uno de estas corresponden. Por ejemplo, no se pueden recibir 200 sacos de cemento si el almacén solo tiene capacidad para 180.

Entre las líneas 14 y 32 se realiza lo siguiente para cada recepción del día:

- Líneas 16-21: Si es una recepción importada, la fecha de ingreso de importación debe ser igual a la fecha de recepción asignada.
- Líneas 23-27: Si no es una recepción importada, la fecha asignada de recepción debe ser mayor o igual a la fecha de compra y menor o igual a la fecha límite de necesidad.
- Líneas 29-32: Se verifica la recepción de la totalidad de la necesidad utilizando la matriz de seguimiento presentada en el anexo E de estructuras auxiliares para los algoritmos.

En caso no se cumpla alguna de estas restricciones, el marcador de error se activará y se finaliza el método indicando que no es una posible solución al problema adecuada.

6.2.2 Validaciones realizadas

Se han realizado las pruebas de caja blanca correspondientes para soluciones de tamaño pequeño. Esto permite dar seguimiento a las funcionalidades del algoritmo para verificar si efectivamente favorece a la optimización de las soluciones iniciales. En el anexo G se presenta un ejemplo a detalle.

Este capítulo ha sido revisado por el especialista en algoritmia, quien ha dado su aprobación respecto a la información presentada en el mismo. En el anexo H se adjunta la evidencia.

6.3 Discusión

En este capítulo se ha alcanzado el resultado esperado R2.2, relacionado a la adaptación del pseudocódigo del algoritmo genético para el problema de planificación que se desea resolver.

El alcance de estos resultados permite comprender el funcionamiento general del algoritmo genético, así como el detalle de cada uno de sus operadores principales. Además, se describe cómo se utiliza la estructura del cromosoma presentada en el capítulo anterior para generar mejores soluciones. Con este avance se establece una base sólida de diseño que será utilizada para la posterior codificación del algoritmo utilizando las herramientas de programación.

Respecto a investigaciones anteriores y casos de estudio seleccionados en el Estado del Arte, se ha identificado que la mayor parte de estos detalla la adaptación del algoritmo que se va a utilizar en el caso de estudio, así como las modificaciones que algunos investigadores realizan sobre estos con la finalidad de optimizar el rendimiento durante la ejecución.

El diseño presentado se ha definido para resolver el problema de planificación de este proyecto. Mediante revisión del Estado del Arte se escogieron los métodos de selección por ruleta y casamiento por recombinación uniforme; sin embargo, es posible modificar estos procedimientos seleccionando alguna de las numerosas técnicas que existen. Por ejemplo, se puede aplicar selección por torneo y casamiento por un solo punto de corte.

Finalmente, se debe indicar que no se realizará una calibración de variables para el algoritmo genético, ya que se seleccionarán los valores promedio en la literatura. Por ejemplo, la tasa de casamiento suele ser 0.5 y la tasa de mutación 0.1.

Capítulo 7. Diseño del algoritmo Cuckoo Search

7.1 Introducción

En este capítulo se desarrolla el resultado esperado R3.2, relacionado al diseño del algoritmo Cuckoo Search adaptado para resolver el problema de planificación. Se presenta el pseudocódigo general, así como los métodos principales del algoritmo, tales como la modificación de nidos por distribución de Lévy y el reemplazo de los nidos menos favorables al final de cada iteración.

7.2 Resultados alcanzados

En este apartado se presenta el detalle de lo realizado para alcanzar el resultado esperado R3.2: "Algoritmo Cuckoo Search diseñado para resolver el problema de planificación".

7.2.1 Algoritmo Cuckoo Search diseñado para resolver el problema de planificación.

La presentación del diseño del algoritmo Cuckoo Search está dividida en los métodos principales que lo componen, con la finalidad de poder explicar el procedimiento de cada uno por separado.

7.2.1.1 Pseudocódigo del algoritmo Cuckoo Search

```
INICIO AlgoritmoCuckooSearch(numNidos, porcDescubrir)
    nidos <- construirPoblacionInicial(numNidos)
    MIENTRAS condicionParada(tiempoMaximo,maxGeneraciones) HACER
        nido1 <- seleccionAleatoria(nidos)
        nuevoNido1 <- actualizarPorVuelodeLevy(nido1)
        SI EsValido(nuevoNido1) ENTONCES
            nido2 <- seleccionAleatoria(nidos)
            Si fitness(nuevoNido1) < fitness(nido2) ENTONCES
                nido2 <- nuevoNido1
            FIN SI
        FIN SI
        construirNuevosNidos(nidos, porcDescubrir)
        ordenarAscFitness(nidos)
    FIN MIENTRAS
    mejorSolucion <- obtenerMejorSolucion(nidos)
FIN AlgoritmoCuckooSearch
```

Figura 18. Pseudocódigo del algoritmo Cuckoo Search. Fuente: Elaboración propia.

En la Figura 18 se muestra el pseudocódigo principal del algoritmo Cuckoo Search, el cual recibe dos parámetros importantes para el algoritmo: el número de nidos a utilizar y el porcentaje de nidos descubiertos para cada iteración.

Para el procedimiento presentado se genera una población aleatoria de nidos en primer lugar, para la cual se valida que cada solución que la conforma cumple con las restricciones del problema (línea 2). Luego, mientras alguna condición de parada no se cumpla (línea 3), se realiza la selección aleatoria de un nido (línea 4) y se hace una modificación del mismo siguiendo una distribución de Lévy (línea 5). A continuación, si el nido es válido (línea 6), se compara el nuevo valor fitness de este respecto a otro seleccionado aleatoriamente (línea 8). En caso de que sea menor, ya que la función objetivo es de minimización, se reemplaza por el recientemente modificado (línea 9). Finalmente, se construyen nuevos nidos a partir de los descubiertos, se realiza un ordenamiento ascendente por valor de fitness y se repite el procedimiento descrito.

7.2.1.2 Construcción de la población inicial

```
1  INICIO construirPoblacionInicial(numNidos)
2      longNido <- diasPlanificacion*horasDia;
3      nido[longNido] <- inicializar
4      actualizarFechaLimiteMargSeguridad(nido)
5      PARA cont=1 HASTA numVehiculos HACER
6          SI esRecepcionImportada(vehiculo) ENTONCES
7              priorizarVehiculo(vehiculo)
8          SI NO
9              diaInicio <- fechaCompraRecepcion
10             diaFin <- fechaLimiteRecepcion
11             PARA i=diaFin DECRECE HASTA diaInicio HACER
12                 SI validarCapacidadAlmacen(i, unidadRecepcion, cantRecepcion) ENTONCES
13                     asignarVehiculo(vehiculo)
14                 FIN SI
15             FIN PARA
16         FIN SI
17     FIN PARA
18 FIN construirPoblacionInicial
```

Figura 19. Pseudocódigo para la construcción de un nido en el algoritmo Cuckoo Search. Fuente: Elaboración propia.

En la Figura 19 se muestra el proceso de construcción de una solución al problema. Este se repite las veces necesarias hasta alcanzar el tamaño deseado de población inicial. Como se puede observar, y dado que ambos algoritmos comparten la estructura principal de solución, la forma de construirla es la misma respecto al algoritmo genético.

Primero, se inicializa la estructura del nido según el horizonte de planificación deseado y las horas de trabajo diarias (líneas 2 y 3). Luego, se actualiza la fecha límite de recepción para cada materia considerando el margen de seguridad (línea 4). A continuación, se recorren todos los vehículos que deben entregar una recepción (línea 5) y, para cada uno, se verifica, en primer lugar, si este tiene asociada una recepción importada (línea 6). En caso de que sea así, se debe priorizar al vehículo, ya que esta debe ser asignada a la fecha de ingreso de importación (7). En esta función se realizará la asignación prioritaria de la recepción a la fecha deseada, incluso reasignando otros vehículos a fechas distintas de ser necesario. Por otro lado, si no es una recepción prioritaria, se evalúa, para el rango permitido de fechas de la recepción, la asignación del vehículo a alguno de estos días (línea 13), verificando previamente la capacidad del almacén que será el encargado de recepcionar la materia prima (línea 12). En esta función de asignación, a diferencia de la que se encarga de recepciones prioritarias, se validará para cada día de la iteración, si existe algún espacio de tiempo disponible para asignar este vehículo. En el caso de que todas las horas del día estén ocupadas, se realiza el mismo procedimiento para los siguientes.

7.2.1.3 Actualización de nidos por vuelos de Lévy

```
1  INICIO actualizarPorVuelodeLevy(nido)
2      tamañoNido <- nido.longitud
3      beta <- 1.5
4      sigma <- 0.6966
5      cotaInf <- 0
6      cotaSup <- tamañoNido - 1
7      PARA i=0 HASTA tamañoNido-1 HACER
8          u <- aleatorio(0,tamañoNido)*sigma
9          v <- aleatorio(0,tamañoNido)
10         paso <- u / potencia(absoluto(v), 1/beta)
11         tamañoPaso <- 0.1*paso*(cotaSup-i)
12         posActual <- i
13         posNuevo <- i + aleatorio(0, tamañoNido)*tamañoPaso
14         posNuevo <- acotar(posNuevo, cotaInf, cotaSup)
15         intercambiar(nido, posActual, posNuevo)
16     FIN PARA
17     retornar(nido)
18 FIN actualizarPorVuelodeLevy
```

Figura 20. Pseudocódigo de la actualización de nido por distribución de Lévy en el algoritmo Cuckoo Search.

Fuente: Elaboración propia.

En la Figura 20 se muestra el proceso para actualizar un nido siguiendo una distribución de Lévy. Es posible utilizar diversas funciones de probabilidad para realizar esta tarea, pero para este proyecto se ha seleccionado el Algoritmo de Mantegna, el cual es el mismo método que utiliza el autor del algoritmo Cuckoo Search en su obra "Algoritmos metaheurísticos bio-inspirados".

Se calcula el tamaño del nido como primer paso (línea 2) y se inicializan las variables importantes para calcular la distribución de Lévy (líneas 3-6). Luego, para cada elemento del nido, se define el tamaño del paso que este va a realizar en su desplazamiento (líneas 8-11). Finalmente, se realiza un intercambio de elementos entre la posición actual de la iteración y la nueva posición resultante (líneas 13-15), retornando el nido modificado al final del proceso.

Las variables "beta" y "sigma" se han obtenido mediante revisión de la literatura para el algoritmo Cuckoo Search, verificando en las diversas aplicaciones del mismo que se toma el

valor de 1.5 para “beta”, mientras que “sigma” se calcula utilizando la fórmula mostrada en la Figura 21, resultando en un valor aproximado de 0.6966.

$$\sigma_u = \left\{ \frac{\Gamma(1+\beta)\sin(\pi\beta/2)}{\Gamma[(1+\beta)/2]\ \beta\ 2^{(\beta-1)/2}} \right\}^{1/\beta}$$

Figura 21. Fórmula para el cálculo de la variable “sigma” en el Algoritmo de Mantegna. Adaptado de (Yang, 2010).

7.2.1.4 Reemplazo de nidos menos favorables

```
1 INICIO construirNuevosNidos(nidos, porcDescubrir)
2     numDescubiertos = numNidos*porcDescubrir
3     ordenarDescFitness(nidos)
4     PARA i=1 HASTA numDescubiertos HACER
5         nidos[i] <- crearNuevaSolucion()
6     FIN PARA
7     retornar(nidos)
8 FIN construirNuevosNidos
```

Figura 22. Pseudocódigo de reemplazo de nidos menos favorables en el algoritmo Cuckoo Search. Fuente: Elaboración propia.

En la Figura 22 se detalla el procedimiento para construir nuevos nidos reemplazando a los que fueron descubiertos. Este se realiza considerando el porcentaje presentado en el pseudocódigo principal.

En primer lugar, se define la cantidad de nidos que serán descubiertos (línea 2). Luego, se ordenan los nidos de manera descendente; es decir, el de mayor valor fitness al inicio (línea 3). Después, para cada uno de estos, se crea una nueva solución aleatoria, la cual va a reemplazar a este nido no favorable, verificando previamente si cumple las restricciones del problema

(líneas 4 y 5). Finalmente, se retorna la nueva población de nidos que sirve como punto de partida para la siguiente iteración (línea 7).

7.2.1.5 Control de aberraciones

```
1  INICIO esValido(nido)
2      flagError <- 0
3      PARA i=1 hasta díasPlanificacion HACER
4          sumaAlmacenTipo <- 0
5          //Capacidad de almacenamiento
6          PARA tipoMP=1 HASTA totalTiposMP HACER
7              sumaAlmacenTipo <- sumatoriaRecepcionesTipo(cromosoma)
8              SI(sumaAlmacenTipo > capacidadMaxDiai) ENTONCES
9                  flagError <- 1
10                 SALIR
11             FIN SI
12         FIN PARA
13         //Recepciones
14         PARA recepcion=1 hasta totalRecepcionesDia HACER
15             //Fecha de importacion igual a fecha de recepcion
16             SI(esRecepcionImportada(vehiculo)) ENTONCES
17                 SI(i != fechaImportacion) ENTONCES
18                     flagError <- 1
19                     SALIR
20                 FIN SI
21             SI NO
22             //Fecha de recepcion entre fecha de compra y fecha limite
23                 SI(fechaCompraRecepcion > i) O (fechaLimiteRecepcion < i) ENTONCES
24                     flagError <- 1
25                     SALIR
26                 FIN SI
27             FIN SI
28             //Se recibe la totalidad de la solicitud en matriz de seguimiento
29             SI noCumpleTotalidad(recepcion) ENTONCES
30                 flagError <- 1
31                 SALIR
32             FIN SI
33         FIN PARA
34     FIN PARA
35 FIN esValido
```

Figura 23. Pseudocódigo del control de aberraciones para el algoritmo Cuckoo Search. Fuente: Elaboración propia.

En la Figura 23 se presenta el pseudocódigo para el control de aberraciones en las posibles soluciones al problema. Dado que ambos algoritmos de este proyecto comparten la estructura de solución, el procedimiento para validar el cumplimiento de las restricciones es el mismo.

En este caso, para el algoritmo Cuckoo Search, se utiliza con la finalidad de garantizar la calidad de los nidos que se utilizan durante la ejecución.

Primero, se inicializa un marcador de error (línea 2), el cual se utiliza como indicador de que se cumplen o no las restricciones al finalizar el método. Luego, se itera para cada día de planificación (línea 3) y en cada uno de estos se verifica el límite de capacidad de almacenamiento. También, para cada recepción del día que está siendo evaluado, se validan las fechas de recepción y el cumplimiento de la recepción de la totalidad de la necesidad.

Entre las líneas 6 y 12 se verifica que las recepciones del día no hayan excedido la capacidad del almacén al que cada uno de estas corresponden. Por ejemplo, no se pueden recibir 200 sacos de cemento si el almacén solo tiene capacidad para 180.

Entre las líneas 14 y 32 se realiza lo siguiente para cada recepción del día:

- Líneas 16-21: Si es una recepción importada, la fecha de ingreso de importación debe ser igual a la fecha de recepción asignada.
- Líneas 23-27: Si no es una recepción importada, la fecha asignada de recepción debe ser mayor o igual a la fecha de compra y menor o igual a la fecha límite de necesidad.
- Líneas 29-32: Se verifica la recepción de la totalidad de la necesidad utilizando la matriz de seguimiento presentada en el anexo E de estructuras auxiliares para los algoritmos.

En caso no se cumpla alguna de estas restricciones, el marcador de error se activará y se finaliza el método indicando que no es una posible solución al problema adecuada.

7.2.2 Validaciones realizadas

Se han realizado las pruebas de caja blanca correspondientes para soluciones de tamaño pequeño. Esto permite dar seguimiento a las funcionalidades del algoritmo para verificar si

efectivamente favorece a la optimización de las soluciones iniciales. En el anexo I se presenta un ejemplo a detalle.

Este capítulo ha sido revisado por el especialista en algoritmia, quien ha dado su aprobación respecto a la información presentada en el mismo. En el anexo J se adjunta la evidencia.

7.3 Discusión

En este capítulo se ha alcanzado el resultado esperado R3.2, relacionado al diseño del pseudocódigo del algoritmo Cuckoo Search para el problema de planificación que se desea resolver.

El alcance de estos resultados permite comprender el funcionamiento general del algoritmo Cuckoo Search, así como el detalle de cada una de sus principales funcionalidades. Además, se describe cómo se utiliza la estructura del nido presentada en el capítulo anterior para generar mejores soluciones. Con este avance se establece una base sólida de diseño que será utilizada para la posterior codificación del algoritmo utilizando las herramientas de programación.

Respecto a investigaciones anteriores y casos de estudio seleccionados en el Estado del Arte, se ha identificado que la mayor parte de estos detalla la adaptación del algoritmo que se va a utilizar en el caso de estudio, así como las modificaciones que algunos investigadores realizan sobre estos con la finalidad de optimizar el rendimiento durante la ejecución.

El diseño presentado se ha definido para resolver el problema de planificación de este proyecto. Sin embargo, es posible modificar, por ejemplo, el proceso de actualización de nidos utilizando una distribución de Gauss u otra técnica, según el rendimiento que se espera obtener del algoritmo. También, se puede modificar la forma de remover los nidos menos favorables construyendo otras soluciones a partir de estos, utilizando variables establecidas por el autor para este proceso. De esta manera, se puede evaluar el comportamiento del algoritmo en diferentes escenarios y para los fines que sea requerido.

Finalmente, se debe indicar que, a diferencia del algoritmo genético, para el algoritmo Cuckoo Search sí se realizará una calibración de variables, puesto que se espera obtener el mejor rendimiento posible durante la ejecución del mismo.

Capítulo 8. Codificación del algoritmo genético

8.1 Introducción

En este capítulo se desarrolla el resultado 4.1, el cual está relacionado al desarrollo de código para el algoritmo genético. La codificación se realizó tomando como base los diseños previamente realizados: el pseudocódigo del algoritmo, la estructura del cromosoma y las estructuras auxiliares de datos.

Se presentarán, al igual que en el diseño del algoritmo, los métodos principales del mismo, considerando esta vez aspectos técnicos básicos y anotaciones sobre situaciones presentadas en el proceso de desarrollo.

En el anexo N se encuentra la dirección URL para acceder al código del algoritmo genético.

8.2 Resultados alcanzados

En este apartado se presenta el detalle de lo realizado para alcanzar el resultado esperado R4.1: "Codificación del algoritmo genético adaptado al problema de planificación".

8.2.1 Codificación del algoritmo genético adaptado al problema de planificación.

Al igual que en la etapa de diseño, la presentación de la codificación del algoritmo genético se divide en los métodos principales que lo componen, con el objetivo de poder explicar lo realizado en cada uno de estos.

8.2.1.1 Método principal del algoritmo genético

```
while ((contadorGeneraciones < maxGeneraciones) && (sinMejora < 1000) && ((tiempoFin - tiempoInicio) < (minutosMax * 1000))) {
    //CASAMIENTO
    poblacion.casamiento(tasaCasamiento);
    //MUTACION
    poblacion.mutacion(tasaMutacion);
    //DEPURACION
      poblacion.imprimirPoblacionGenerada();
    poblacion = poblacion.depurar(tamanhoPoblacion, porcConservacion);
      poblacion.imprimirPoblacionGenerada();
    //VALIDAR MEJORA
    Cromosoma mejorActual = poblacion.obtenerMejorSolucion(0);
    if (mejorSolucion.getValorFitness() > mejorActual.getValorFitness()) {
        mejorSolucion = mejorActual;
        sinMejora = 0;
    } else {
        sinMejora++;
    }
    contadorGeneraciones++;
    tiempoFin = System.currentTimeMillis();
}
```

Figura 24. Codificación de método principal para el algoritmo genético. Fuente: Elaboración propia.

En la Figura 24 se puede observar la codificación del método principal para el algoritmo genético, en el cual se consideró el número de generaciones y el tiempo máximo de ejecución, variables establecidas en la etapa de diseño. Además, se agregó una nueva variable que cuenta el número de generaciones en las que no se ha presentado una mejora. Esto evitará que el algoritmo siga ejecutándose cuando ya se tiene un número considerable de las mismas sin una evolución satisfactoria para el problema.

8.2.1.2 Construcción de la población inicial

En la Figura 25 se presenta la forma de generar una posible solución al problema. En primer lugar, se realiza la creación de las principales estructuras de datos tales como los almacenes, las recepciones y los vehículos. La bandera que se observa con el nombre "flagInicio" evita que los contenidos estáticos vuelvan a crearse, ya que estos se comparten entre todas las soluciones. Por ejemplo, siempre se trabaja con los mismos almacenes. Sin embargo, existen otras estructuras, como la matriz de capacidades de almacén y la matriz de seguimiento de recepciones, las cuales sí deben ser inicializadas cada vez que se genera una posible solución al problema, ya que estas son actualizadas constantemente durante la evolución de la misma.

Una vez creadas las estructuras necesarias, se procede a realizar la programación de las recepciones, priorizando las importaciones respecto a las nacionales, ya que estas tienen una fecha restrictiva en la que es obligatorio cumplir con su recepción. Finalmente, se realiza la validación de la posible solución generada y se añade a la población inicial de ser el caso. Este procedimiento se repite hasta completar el tamaño deseado.

```
public Cromosoma construir(int diasPlanificacion, int horasDia) {
    this.getGestorAlmacen().crearAlmacenes(diasPlanificacion, flagInicio);
    this.getGestorRecepcion().crearRecepciones(diasPlanificacion, horasDia, this.gestorAlmacen, flagInicio);
    this.getGestorVehiculo().crearVehiculos(this.gestorRecepcion, flagInicio);
    //GENERAR PROGRAMACION
    //IMPORTACIONES TIENEN PRIORIDAD
    asignarEspacioImportaciones();
    //LUEGO NACIONALES
    asignarEspacioNacionales();
    //VALIDAR CROMOSOMA
    flagInicio = true;
    if (esValido()) {
        calcularFitness();
        return this;
    } else {
        return null;
    }
}
```

Figura 25. Codificación de la generación de población inicial para el algoritmo genético. Fuente: Elaboración propia.

8.2.1.3 Método de selección

En la Figura 26 se puede observar el método de selección codificado, el cual corresponde a la técnica de la ruleta diseñada previamente. Este módulo considera la modificación de invertir la probabilidad respecto a la proporción de valor fitness; es decir, a menor valor de adaptación, mayor será el espacio asignado en la ruleta.

Se agregó una consideración respecto a este módulo, ya que se detectaron situaciones en las que, al generarse una probabilidad aleatoria muy cercana a uno, la ruleta no seleccionaba ningún individuo debido a la precisión de los números decimales. Se optó por devolver un individuo aleatorio cuando se presente esta situación.

```java
public Cromosoma seleccionarRuleta() {
    double suma = 0.0;
    Random rand = new Random();
    double probAleatoria = rand.nextDouble();
    for (int i = 0; i < this.tamanhoPoblacion; i++) {
        double fitnessCromosoma = this.individuos.get(i).getValorFitness();
        suma += (double) (1 - fitnessCromosoma / this.totalFitness) / (this.tamanhoPoblacion - 1);
        if (suma >= probAleatoria) {
            return this.individuos.get(i);
        }
    }
    //SI NO SE SELECCIONA NINGUNO POR RULETA SE DEVUELVE UN RANDOM
    return obtenerMejorSolucion(rand.nextInt(this.tamanhoPoblacion));
}
```

Figura 26. Codificación del método de selección para el algoritmo genético. Fuente: Elaboración propia.

8.2.1.4 Método de casamiento

```java
public void casamiento(double tasaCasamiento) {
    int numCasamientos = (int) Math.ceil(this.tamanhoPoblacion * tasaCasamiento);
    for (int i = 0; i < numCasamientos; i++) {
        Cromosoma padre1 = seleccionarRuleta();
        Cromosoma padre2 = seleccionarRuleta();
        Cromosoma hijo1 = padre1.recombinacion(padre2, tasaCasamiento);
        if (hijo1.esValido()) {
            hijo1.calcularFitness();
            agregarSolucion(hijo1);
        }
        Cromosoma hijo2 = padre2.recombinacion(padre1, tasaCasamiento);
        System.out.println("HIJO2:");
        hijo2.imprimirCromosoma();
        if (hijo2.esValido()) {
            hijo2.calcularFitness();
            agregarSolucion(hijo2);
        }
    }
}
```

Figura 27. Codificación del método de casamiento para el algoritmo genético. Fuente: Elaboración propia.

En el método de casamiento, mostrado en la Figura 27, no se presentan diferencias relevantes u optimizaciones realizadas respecto a la etapa de diseño. El procedimiento de seleccionar dos padres y crear dos hijos a partir de estos, los cuales son validados antes de ser agregados a la población, se mantiene en su forma base.

Es importante recalcar que, antes que se genere cualquiera de los dos hijos mediante el casamiento, se verifica internamente si el gen afecto al posible reemplazo es una importación. En este caso, y por lógica de negocio, no se realiza la modificación, ya que estas recepciones deben estar obligatoriamente presentes en las fechas definidas.

Se muestra a continuación, en la Figura 28, un resultado del método de casamiento ejecutado mediante código. Como se puede observar, el segundo hijo generado toma genes tanto del primer padre como del segundo, según valores de probabilidad aleatorios que se comparan contra la tasa de casamiento.

```
PADRE1:
X    MP0  X    MP0  MP1  MP3  MP3  MP2  X    MP4  MP4  MP2  MP6  MP6  X    MP5
PADRE2:
MP0  X    MP0  X    MP1  MP3  MP3  X    MP2  MP2  MP4  MP4  MP6  MP6  X    MP5
HIJO2:
MP0  X    MP0  X    MP1  MP3  MP3  MP2  X    MP2  MP4  MP4  MP6  MP6  X    MP5
```

Figura 28. Ejemplo de casamiento para dos cromosomas en el algoritmo genético. Fuente: Elaboración propia.

8.2.1.5 Método de mutación

```
public void mutacion(double tasaMutacion) {
    int numMutaciones = (int) Math.ceil(this.tamanhoPoblacion * tasaMutacion);
    for (int i = 0; i < numMutaciones; i++) {
        Random rand = new Random();
        int posAleatoria = rand.nextInt(this.tamanhoPoblacion);
        Cromosoma solucionAMutar = this.individuos.get(posAleatoria);
        Cromosoma solucionMutada = solucionAMutar.mutar();
        if (solucionMutada.esValido()) {
            solucionMutada.calcularFitness();
            agregarSolucion(solucionMutada);
        }
    }
}
```

Figura 29. Codificación del método de mutación para el algoritmo genético. Fuente: Elaboración propia.

En la Figura 29 se presenta el método de mutación, el cual, mantiene también el diseño realizado previamente. En este módulo se presentó un inconveniente técnico relacionado a la forma de trabajo de Java con los objetos. Cuando se modificaban los genes tomados de la solución a mutar en la solución mutada, se sobrescribía este en la solución a mutar, sea o no válido. Esto ocurre debido a que Java mantiene la referencia a la posición de memoria en la que este arreglo se encuentra. Mediante depuración del código se solucionó este problema creando una copia de los genes antes de empezar a mutarlos, tal que la referencia en memoria sea distinta para cada objeto.

De igual forma que en el casamiento, antes de intercambiar dos genes del cromosoma para la mutación, se verifica internamente si alguno de estos es una importación. De ser el caso, no se realiza el cambio de posiciones y se continúa con la siguiente.

Se muestra, en la Figura 30, un resultado del método de mutación ejecutado mediante código. En este caso, la solución a mutar se ve afecta a diferentes intercambios aleatorios de genes, con la finalidad de obtener una variación que pueda favorecer la exploración del espacio de búsqueda, adicional al método de casamiento.

```
SOLUCION A MUTAR
X    X    MP3  X    MP4  X    MP1  MP6  MP5  MP5  MP0  MP3  MP2  MP2  X    X
SOLUCION MUTADA
MP1  X    MP3  X    MP4  MP3  MP0  MP6  MP5  MP5  X    X    MP2  MP2  X    X
```

Figura 30. Ejemplo de mutación de un cromosoma para el algoritmo genético. Fuente: Elaboración propia.

8.2.1.6 Evaluación de la población

```
public Poblacion depurar(int tamanhoOriginal, double porcConservacion) {
    int cantConservar = (int) Math.ceil(this.tamanhoPoblacion * porcConservacion);
    Poblacion nuevaPoblacion = new Poblacion(0);
    for (int i = 0; i < cantConservar; i++) {
        Cromosoma solucion = obtenerMejorSolucion(i);
        nuevaPoblacion.agregarSolucion(solucion);
    }
    for (int i = cantConservar; i < tamanhoOriginal; i++) {
        Cromosoma solucion = seleccionarRuleta();
        nuevaPoblacion.agregarSolucion(solucion);
    }
    nuevaPoblacion.calcularFitnessPoblacion();
    return nuevaPoblacion;
}
```

Figura 31. Codificación del método de depuración de la población en el algoritmo genético. Fuente: Elaboración propia.

Para la evaluación de la población y retorno a la cantidad original de individuos, se utiliza el método presentado en la Figura 31. En este se define una cantidad de cromosomas a conservar los cuales pasan directamente a la siguiente generación. Posteriormente, mediante la técnica de la ruleta se seleccionan a los restantes hasta completar el tamaño de población original que debe tener el inicio de la siguiente generación.

Respecto al diseño, en este método no se presentaron diferencias u optimizaciones considerables al realizar la codificación del mismo.

8.2.1.7 Control de aberraciones

El control de aberraciones se presenta en dos figuras, 32 y 33, ya que es un procedimiento complejo para validar las diversas restricciones que deben cumplir los cromosomas obtenidos durante la generación de la población inicial y la evolución de las generaciones, tanto los hijos como los mutados.

```
public boolean esValido() {
    boolean esValido = true;
    //CAPACIDAD DE ALMACEN
    for (int i = 0; i < this.diasPlanificacion; i++) {
        int cantAlmacenes = this.gestorAlmacen.getAlmacenes().size();
        for (int j = 0; j < cantAlmacenes; j++) {
            int sumaDiaAlmacen = 0;
            String unidadAlmacen = this.gestorAlmacen.getAlmacenes().get(j).getUnidadMP();
            for (int k = this.horasDia * i; k < this.horasDia * (i + 1); k++) {
                Vehiculo vehiculo = this.genes[k];
                if (vehiculo != null) {
                    Recepcion recepcion = vehiculo.getRecepcion();
                    //FECHAS
                    //IMPORTACION
                    if (recepcion.esImportado()) {
                        int fechaImportacion = recepcion.getFechaImportacion();
                        if (fechaImportacion != i) {
                            esValido = false;
                            return esValido;
                        }
                    } else {
                        //NO ES IMPORTACION
                        int fechaCompra = recepcion.getFechaCompra();
                        int fechaLimite = recepcion.getFechaLimite();
                        if (i < fechaCompra || i > fechaLimite) {
                            esValido = false;
                            return esValido;
                        }
                    }
```

Figura 32. Codificación del control de aberraciones para el algoritmo genético. Fuente: Elaboración propia.

En la primera parte (Figura 32), tomando como punto de partida el comentario “Capacidad de almacén”, se realiza un recorrido para todos los días de planificación estimados. Luego, para cada almacén, se hace un recorrido de las horas del día.

También, como se puede observar en los comentarios “Importación” y “No es importación”, se realiza por cada vehículo encontrado en el día, la verificación de la recepción asociada. Si es una importación, la fecha de recepción asignada debe ser el mismo día de ingreso al país. De no serlo, se verifica que se haya programado su recepción dentro del plazo establecido, entre la fecha de compra y la fecha límite de necesidad.

A continuación, en la segunda parte del procedimiento (Figura 33), y dentro de la iteración comentada anteriormente para cada almacén, se va realizando la acumulación de capacidad de almacenamiento utilizada con la variable “sumaDiaAlmacen”. Al terminar la iteración de un determinado almacén y del recorrido de las horas del día, este valor se compara contra la

capacidad disponible inicialmente en el día, la cual se encuentra en una matriz diferente a la de actualización de capacidades, para validar si no ha ocurrido una aberración.

Finalmente, una vez terminado el seguimiento a cada recepción del cromosoma evaluado y utilizando la matriz de seguimiento de las mismas, se verifica si se ha cumplido con recepcionar la totalidad de la necesidad para cada una. En caso de que se haya recibido menos (o más, por alguna evolución incorrecta) de alguna materia prima, se lanzará una excepción, indicando de que la posible solución no es válida para el problema.

```
                    } else {
                        //NO ES IMPORTACION
                        int fechaCompra = recepcion.getFechaCompra();
                        int fechaLimite = recepcion.getFechaLimite();
                        if (i < fechaCompra || i > fechaLimite) {
                            esValido = false;
                            return esValido;
                        }
                    }
                    if (recepcion.getUnidadMP().equals(unidadAlmacen)) {
                        sumaDiaAlmacen += vehiculo.getCantTransporte();
                    }
                }
            }
            //QUE NO SUPERE LA CAPACIDAD
            if (sumaDiaAlmacen > GestorAlmacen.getMatrizBase().get(j).get(i)) {
                esValido = false;
                return esValido;
            }
        }
    }
    //VERIFICAR MATRIZ DE SEGUIMIENTO
    for (int i = 0; i < this.gestorRecepcion.getMatrizSeguimiento().size(); i++) {
        int solicitado = this.gestorRecepcion.getMatrizSeguimiento().get(i).get(0);
        int actual = this.gestorRecepcion.getMatrizSeguimiento().get(i).get(1);
        if (solicitado != actual) {
            esValido = false;
            return esValido;
        }
    }
    return esValido;
```

Figura 33. Codificación del control de aberraciones para el algoritmo genético. Fuente: Elaboración propia.

8.2.2 Validaciones realizadas

Se realizaron diversas pruebas para poblaciones pequeñas, con la finalidad de evaluar el funcionamiento de cada uno de los operadores principales del algoritmo, así como la evolución correcta de soluciones y validación de las mismas. En el anexo K se presenta, a detalle, una prueba realizada a la codificación del algoritmo.

8.3 Discusión

En este capítulo se ha alcanzado el resultado esperado R4.1, relacionado a la codificación del algoritmo genético para el problema de planificación que se desea resolver, basado en el diseño previamente realizado tanto de pseudocódigo como de estructuras principales y de soporte.

El alcance de estos resultados permite contar con el algoritmo genético desarrollado y operativo. Además, al tener el código implementado y observar las situaciones presentadas durante la ejecución del mismo, es posible conocer con mayor detalle cómo trabaja este algoritmo metaheurístico y la forma en que realiza la evolución de individuos. También, enfrenta al desarrollador a diversos retos como la comprensión del lenguaje de programación utilizado y la corrección de problemas presentados por conceptos propios del mismo. Con este avance se establece un paso importante respecto a la obtención de resultados del proyecto y se evidencia la relevancia de contar con un diseño previo, ya que favorece en gran medida la implementación de código.

Finalmente, se debe indicar que, temporalmente, se ha establecido un método para generar posibles soluciones aleatorias al problema, lo que involucra crear tanto almacenes, recepciones y vehículos de la misma forma. Puede darse el caso en el que, por la misma aleatoriedad, se generen datos base que no permitan generar una población inicial, lo que no significa que el algoritmo no esté funcionando correctamente. Más adelante, cuando se cuente con información validada adecuadamente (por ejemplo, desde un archivo) no se presentarán estas situaciones.

Capítulo 9. Codificación del algoritmo Cuckoo Search

9.1 Introducción

En este capítulo se desarrolla el resultado 4.2, la codificación del algoritmo Cuckoo Search. Este proceso se realizó tomando como base los diseños previamente realizados: el pseudocódigo del algoritmo, la estructura del nido y las estructuras auxiliares de datos.

Se presentarán, al igual que en el diseño del algoritmo, los métodos principales del mismo, considerando esta vez aspectos técnicos básicos y anotaciones sobre situaciones presentadas en el proceso de desarrollo.

En el anexo N se encuentra la dirección URL para acceder al código del algoritmo Cuckoo Search.

9.2 Resultados alcanzados

En este apartado se presenta el detalle de lo realizado para alcanzar el resultado esperado R4.2: "Codificación del algoritmo Cuckoo Search diseñado para resolver el problema de planificación".

9.2.1 Codificación del algoritmo Cuckoo Search diseñado para resolver el problema de planificación.

Al igual que en la etapa de diseño, la presentación de la codificación del algoritmo Cuckoo Search se divide en los métodos principales que lo componen, con el objetivo de poder explicar lo realizado en cada uno de estos.

9.2.1.1 Método principal del algoritmo Cuckoo Search

```
while ((contadorGeneraciones < maxGeneraciones) && (sinMejora < 1000) && ((tiempoFin - tiempoInicio) < (minutosMax * 60 * 1000))) {
    //OBTENER UN NIDO RANDOM
    Random rand = new Random();
    int posAleatoria1 = rand.nextInt(tamanhoPoblacion);
    Nido nido1 = poblacion.getIndividuo(posAleatoria1);
    //ACTUALIZAR NIDO POR LEVY
    Nido nuevoNido1 = nido1.modificacionPorLevy(diasPlanificacion, horasDia);
    if (nuevoNido1.esValido()) {
        nuevoNido1.calcularFitness();
        //OBTENER OTRO NIDO RANDOM
        int posAleatoria2 = rand.nextInt(tamanhoPoblacion);
        Nido nido2 = poblacion.getIndividuo(posAleatoria2);
        //REEMPLAZAR NIDO
        if (nuevoNido1.getValorFitness() < nido2.getValorFitness()) {
            poblacion.reemplazarNidos(posAleatoria2, nuevoNido1);
            sinMejora = 0;
        } else {
            sinMejora++;
        }
    }
    //CONSTRUIR NUEVOS NIDOS
    poblacion.construirNuevosNidos(porcDescubrimiento, diasPlanificacion, horasDia);
    mejorSolucion = poblacion.obtenerMejorSolucion(0);
    contadorGeneraciones++;
    tiempoFin = System.currentTimeMillis();
}
```

Figura 34. Codificación del método principal del algoritmo Cuckoo Search. Fuente: Elaboración propia.

En la Figura 34 se muestra el método principal del algoritmo Cuckoo Search, el cual obedece al diseño previamente realizado para el mismo, con una condición de parada adicional relacionada al número de generaciones en las que no se tiene mejora en las posibles soluciones (meseta). De esta forma se controla el algoritmo para evitar que siga realizado iteraciones cuando no ha podido mejorar el resultado.

9.2.1.2 Construcción de la población inicial

```
public Nido construir(int diasPlanificacion, int horasDia) {
    this.getGestorAlmacen().crearAlmacenes(diasPlanificacion, flagInicio);
    this.getGestorRecepcion().crearRecepciones(diasPlanificacion, horasDia, this.getGestorAlmacen(), flagInicio);
    this.getGestorVehiculo().crearVehiculos(this.getGestorRecepcion(), flagInicio);
    //GENERAR PROGRAMACION
    //IMPORTACIONES TIENEN PRIORIDAD
    asignarEspacioImportaciones();
    //LUEGO NACIONALES
    asignarEspacioNacionales();
    //PRESENTAR ESTRUCTURAS AUXILIARES
    //VALIDAR NIDO
    flagInicio = true;
    if (esValido()) {
        calcularFitness();
        return this;
    } else {
        return null;
    }
}
```

Figura 35. Codificación para la construcción de un nido en el algoritmo Cuckoo Search. Fuente: Elaboración propia.

En la Figura 35 se presenta la forma de generar un nido para el algoritmo Cuckoo Search. Dado que se utiliza la misma estructura para el cromosoma y el nido, el procedimiento para crear estas posibles soluciones al problema lo es también.

En primer lugar, se realiza la creación de las principales estructuras de datos tales como los almacenes, las recepciones y los vehículos. La bandera que se observa con el nombre "flagInicio" evita que los contenidos estáticos vuelvan a crearse, ya que estos se comparten entre todas las soluciones. Por ejemplo, siempre se trabaja con los mismos almacenes. Sin embargo, existen otras estructuras, como la matriz de capacidades de almacén y la matriz de seguimiento de recepciones, las cuales sí deben ser inicializadas cada vez que se genera una posible solución al problema, ya que estas son actualizadas constantemente durante la evolución de la misma.

Una vez creadas las estructuras necesarias, se procede a realizar la programación de las recepciones, priorizando las importaciones respecto a las nacionales, ya que estas tienen una fecha restrictiva en la que es obligatorio cumplir con su recepción. Finalmente, se realiza la validación de la posible solución generada y se añade a la población inicial de ser el caso. Este procedimiento se repite hasta completar el tamaño deseado.

9.2.1.3 Actualización de nidos por vuelos de Lévy

En la Figura 36 se presenta la modificación de nidos por distribución de Lévy. Como se estableció en la etapa de diseño del método, se definen las variables "beta" y "sigma", así como las cotas inferior y superior para poder controlar el tamaño del paso a realizar. Luego, para cada vehículo que conforma el nido, se calcula la posición con la que se realizará el intercambio utilizando el algoritmo de Mantegna.

```java
public Nido modificacionPorLevy(int diasPlanificacion, int horasDia) {
    Nido nidoNuevo = new Nido(diasPlanificacion, horasDia);
    Vehiculo[] vehiculosRotados = this.vehiculos.clone();
    double beta = 1.5;
    double sigma = 0.6966;
    int cotaInf = 0;
    int cotaSup = this.tamanhoSolucion - 1;
    for (int i = 0; i < this.tamanhoSolucion; i++) {
        Random rand = new Random();
        double u = rand.nextGaussian() * sigma;
        double v = rand.nextGaussian();
        double paso = u / Math.pow(Math.abs(v), 1 / beta);
        double tamanhoPaso = 0.1 * paso * (cotaSup - i);
        int posActual = i;
        int posNuevo = (int) (i + rand.nextGaussian() * tamanhoPaso);
        //ACOTAR POSNUEVO
        if (posNuevo > cotaSup) {
            posNuevo = cotaSup;
        } else if (posNuevo < 0) {
            posNuevo = cotaInf;
        }
```

Figura 36. Codificación para la actualización de nido por distribución de Lévy en el algoritmo Cuckoo Search.

Fuente: Elaboración propia.

Finalmente, luego del procedimiento mostrado, se procede con el intercambio en sí. Es importante mencionar que, al igual que en el algoritmo genético, se deben verificar los vehículos que transportan materia prima importada, ya que estas recepciones no pueden modificar su fecha asignada de recepción inicial.

9.2.1.4 Reemplazo de nidos menos favorables

En la Figura 37 se muestra el método de reemplazo y construcción de nuevos nidos. Para esto, se define el número de descubrimientos a realizar en base al porcentaje definido para este fin. Luego, en las posiciones de los peores nidos, se construye uno nuevo, utilizando el mismo método de construcción que se utiliza para la generación de la población inicial.

En este punto, es importante validar que la posible solución generada no sea nula, ya que puede darse el caso en el que, al intentar generar un nuevo nido, no se haya podido encontrar una

combinación que cumpla con las restricciones al primer intento. Por lo tanto, esta creación está dentro de un bucle infinito que solo será detenido cuando la posible solución entregada no sea nula.

```
void construirNuevosNidos(double porcDescubrimiento, int diasPlanificacion, int horasDia) {
    int numDescubiertos = (int) Math.ceil(this.tamanhoPoblacion * porcDescubrimiento);
    Collections.sort(this.individuos, Collections.reverseOrder(Comparator.comparing(Nido::getValorFitness)));
    for (int i = 0; i < numDescubiertos; i++) {
        while (true) {
            Nido nidoNuevo = new Nido(diasPlanificacion, horasDia);
            if (nidoNuevo.construir(diasPlanificacion, horasDia) != null) {
                this.individuos.set(i, nidoNuevo);
                break;
            }
        }
    }
}
```

Figura 37. Codificación de reemplazo de nidos menos favorables en el algoritmo Cuckoo Search. Fuente: Elaboración propia.

9.2.1.5 Control de aberraciones

El control de aberraciones se presenta en las Figuras 38 y 39. Este método permite validar las diversas restricciones que deben cumplir los nidos obtenidos durante la generación de la población inicial y la ejecución de las generaciones.

```java
public boolean esValido() {
    boolean esValido = true;
    //CAPACIDAD DE ALMACEN
    for (int i = 0; i < this.getDiasPlanificacion(); i++) {
        int cantAlmacenes = this.getGestorAlmacen().getAlmacenes().size();
        for (int j = 0; j < cantAlmacenes; j++) {
            int sumaDiaAlmacen = 0;
            String unidadAlmacen = this.getGestorAlmacen().getAlmacenes().get(j).getUnidadMP();
            for (int k = this.getHorasDia() * i; k < this.getHorasDia() * (i + 1); k++) {
                Vehiculo vehiculo = this.vehiculos[k];
                if (vehiculo != null) {
                    Recepcion recepcion = vehiculo.getRecepcion();
                    //FECHAS
                    //IMPORTACION
                    if (recepcion.esImportado()) {
                        int fechaImportacion = recepcion.getFechaImportacion();
                        if (fechaImportacion != i) {
                            esValido = false;
                            return esValido;
                        }
                    } else {
                        //NO ES IMPORTACION
                        int fechaCompra = recepcion.getFechaCompra();
                        int fechaLimite = recepcion.getFechaLimite();
                        if (i < fechaCompra || i > fechaLimite) {
                            esValido = false;
                            return esValido;
                        }
                    }
```

Figura 38. Codificación del control de aberraciones para el algoritmo Cuckoo Search. Fuente: Elaboración propia.

En la Figura 38, tomando como punto de partida el comentario “Capacidad de almacén”, se realiza un recorrido para todos los días de planificación estimados. Luego, para cada almacén, se hace un recorrido de las horas del día.

También, como se puede observar en los comentarios “Importación” y “No es importación”, se realiza por cada vehículo encontrado en el día, la verificación de la recepción asociada. Si es una importación, la fecha de recepción asignada debe ser el mismo día de ingreso al país. De no serlo, se verifica que se haya programado su recepción dentro del plazo establecido, entre la fecha de compra y la fecha límite de necesidad.

```
                    if (recepcion.getUnidadMP().equals(unidadAlmacen)) {
                        sumaDiaAlmacen += vehiculo.getCantTransporte();
                    }
                }
            }
            //QUE NO SUPERE LA CAPACIDAD
            if (sumaDiaAlmacen > GestorAlmacen.getMatrizBase().get(j).get(i)) {
                esValido = false;
                return esValido;
            }
        }
    }
    //VERIFICAR MATRIZ DE SEGUIMIENTO
    for (int i = 0; i < this.getGestorRecepcion().getMatrizSeguimiento().size(); i++) {
        int solicitado = this.getGestorRecepcion().getMatrizSeguimiento().get(i).get(0);
        int actual = this.getGestorRecepcion().getMatrizSeguimiento().get(i).get(1);
        if (solicitado != actual) {
            esValido = false;
            return esValido;
        }
    }
    return esValido;
}
```

Figura 39. Codificación del control de aberraciones para el algoritmo Cuckoo Search. Fuente: Elaboración propia.

A continuación, en la segunda parte del procedimiento, mostrado en la Figura 39, y dentro de la iteración comentada anteriormente para cada almacén, se va realizando la acumulación de capacidad de almacenamiento utilizada con la variable "sumaDiaAlmacen". Al terminar la iteración de un determinado almacén y del recorrido de las horas del día, este valor se compara contra la capacidad disponible inicialmente en el día, la cual se encuentra en una matriz diferente a la de actualización de capacidades, para validar si no ha ocurrido una aberración.

Finalmente, una vez terminado el seguimiento a cada recepción del cromosoma evaluado y utilizando la matriz de seguimiento de las mismas, se verifica si se ha cumplido con recepcionar la totalidad de la necesidad para cada una. En caso de que se haya recibido menos (o más, por alguna evolución incorrecta) de alguna materia prima, se lanzará una excepción, indicando de que la posible solución no es válida para el problema.

9.2.2 Validaciones realizadas

Se realizaron diversas pruebas para poblaciones de diverso tamaño, con la finalidad de evaluar el funcionamiento de cada uno de los métodos principales del algoritmo, así como la evolución correcta de soluciones y el proceso de validación de las restricciones. En el anexo L se presenta, a detalle, una prueba realizada a la codificación del algoritmo Cuckoo Search.

9.3 Discusión

En este capítulo se ha alcanzado el resultado esperado R4.2, relacionado a la codificación del algoritmo Cuckoo Search para el problema de planificación que se desea resolver, basado en el diseño previamente realizado tanto de pseudocódigo como de estructuras principales y de soporte.

El alcance de estos resultados permite contar con el algoritmo Cuckoo Search desarrollado y funcionando correctamente. Además, al tener el código implementado y observar las situaciones presentadas durante la ejecución del mismo, es posible conocer con mayor detalle cómo trabaja este algoritmo metaheurístico y la forma en que realiza la evolución de los nidos que representan las posibles soluciones al problema. Con este avance se puede afirmar que el desarrollo de ambos algoritmos metaheurísticos trabajados en este proyecto están terminados.

Finalmente, al igual que para el algoritmo genético, existe, temporalmente, la limitación de ejecutar el algoritmo Cuckoo Search en base a datos aleatorios, por lo que puede darse el caso en el que, por la misma aleatoriedad, se generen recepciones que no permitan construir una población inicial. Esto no significa que el algoritmo no esté funcionando correctamente, si no que los datos generados no permiten encontrar combinaciones adecuadas para cumplir con las restricciones del problema. Más adelante, cuando se cuente con información consistente (por ejemplo, desde un archivo) no se presentarán estas situaciones.

Capítulo 10. Interfaz gráfica

10.1 Introducción

En este capítulo se desarrolla el resultado 4.3, relacionado al desarrollo de la interfaz gráfica que permite la carga de datos, configuración de los parámetros de los algoritmos y visualización de resultados obtenidos de la ejecución.

En el anexo N se encuentra la dirección URL para acceder al código de la interfaz gráfica para los algoritmos.

10.2 Resultados alcanzados

En este apartado se presenta el detalle de lo realizado para alcanzar el resultado esperado R4.3: "Interfaz gráfica para la ejecución de los algoritmos".

10.2.1 Interfaz gráfica para la ejecución de los algoritmos

El contenido de esta sección se divide por cada una de las pantallas que conforman la interfaz de usuario para la carga de archivos de datos, configuración de los algoritmos y presentación de resultados obtenidos.

10.2.1.1 Pantalla de carga de datos

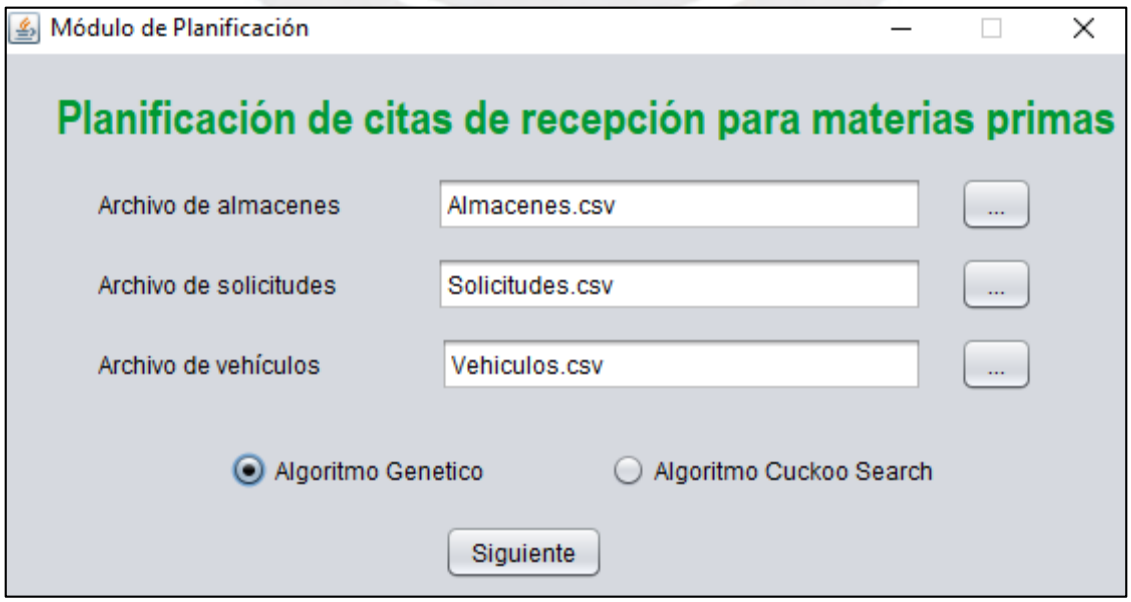


Figura 40. Pantalla de carga de datos. Fuente: Elaboración propia.

En la Figura 40 se muestra la pantalla de carga de datos, la cual permite adjuntar tres archivos ".csv" y seleccionar el algoritmo que se desea ejecutar utilizando estos datos.

El primero de estos archivos es el de almacenes y sus capacidades iniciales diarias. El segundo, el archivo de solicitudes, contiene los datos de las recepciones a realizar, así como la definición de los días de planificación y horas de trabajo diarias. El último, el archivo de vehículos, tiene la placa de identificación, la materia prima y cantidad transportada por cada uno de estos. En el anexo M se muestra el detalle del contenido de estos archivos.

Finalmente, luego de cargar los archivos, se selecciona el algoritmo deseado y se procede a configurarlo en la siguiente pantalla. También es posible seleccionar ambos algoritmos, en caso se desee comparar posteriormente las soluciones obtenidas.

10.2.1.2 Pantalla de configuración de algoritmo genético

En la Figura 41 se presenta la pantalla de configuración del algoritmo genético. En esta vista, se establecen los parámetros para la ejecución del mismo, tales como: tamaño de población, máximo de generaciones, tasa de casamiento, tasa de mutación, tasa de conservación de soluciones (elitismo) y tiempo máximo de ejecución en minutos.

Por defecto, en esta pantalla se presentan valores establecidos para los parámetros mencionados. Estos representan el promedio de los valores más adecuados para los mismos, según diversos estudios y aplicaciones del algoritmo genético.

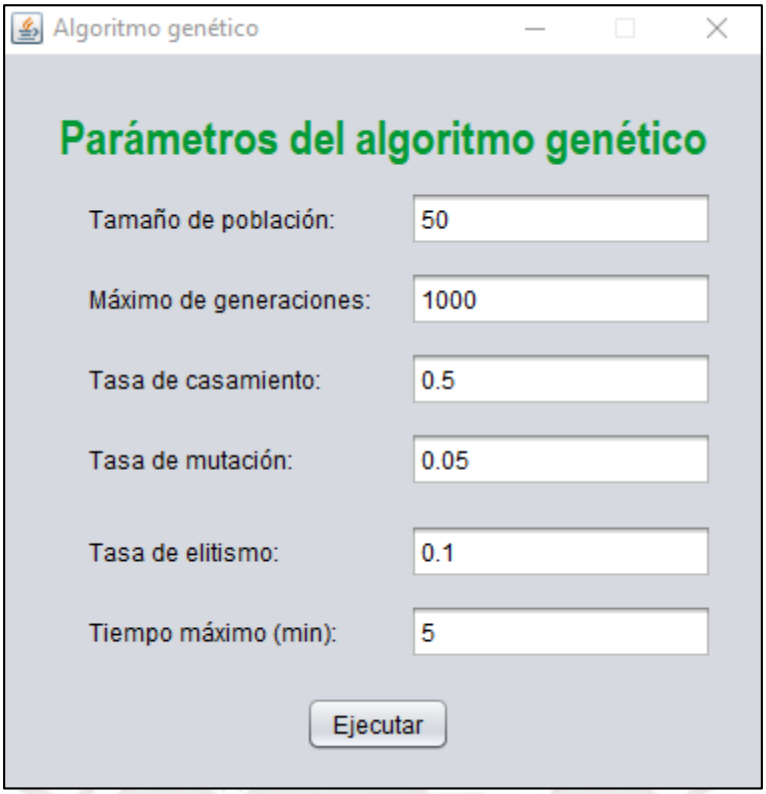


Figura 41. Pantalla de configuración del algoritmo genético. Fuente: Elaboración propia.

10.2.1.3 Pantalla de configuración de algoritmo Cuckoo Search

En la Figura 42 se presenta la pantalla de configuración del algoritmo Cuckoo Search. En esta vista se establecen los parámetros para la ejecución del mismo: tamaño de población, máximo de generaciones, tasa de descubrimiento para los peores nidos y tiempo máximo de ejecución en minutos.

En esta pantalla se presentan valores establecidos por defecto para los parámetros mencionados. Estos representan los valores más adecuados para los mismos, según la calibración realizada para el algoritmo Cuckoo Search, aplicado al problema de planificación estudiado en este proyecto (Ver Capítulo 11).

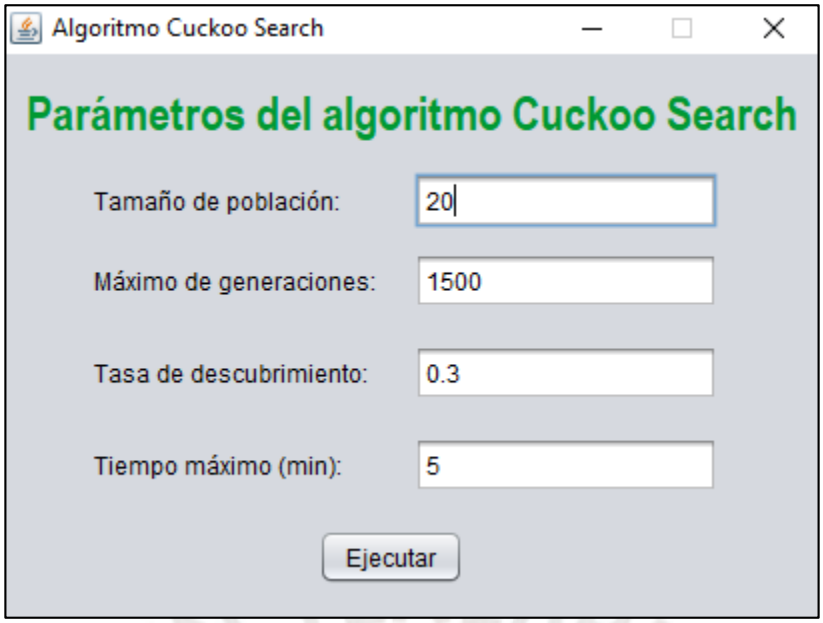


Figura 42. Pantalla de ejecución de algoritmo Cuckoo Search. Fuente: Elaboración propia.

10.2.1.4 Pantalla de configuración de ambos algoritmos

En caso se hayan seleccionado ambos algoritmos para ser ejecutados, la pantalla mostrada será la combinación de las presentadas para configurar cada uno de forma individual.

10.2.1.5 Pantalla de visualización de resultados

En la Figura 43 se muestra la pantalla que permite visualizar los resultados obtenidos de la planificación. Estos estarán ordenados por día y hora.

Las cabeceras detallan el contenido que se presentará al obtener una planificación: identificador de vehículo (placa), nombre de la materia prima asociada, cantidad transportada, día asignado para la recepción y horario de la misma.

Asimismo, se muestra también el valor fitness obtenido para la planificación presentada, según la ejecución del algoritmo correspondiente.

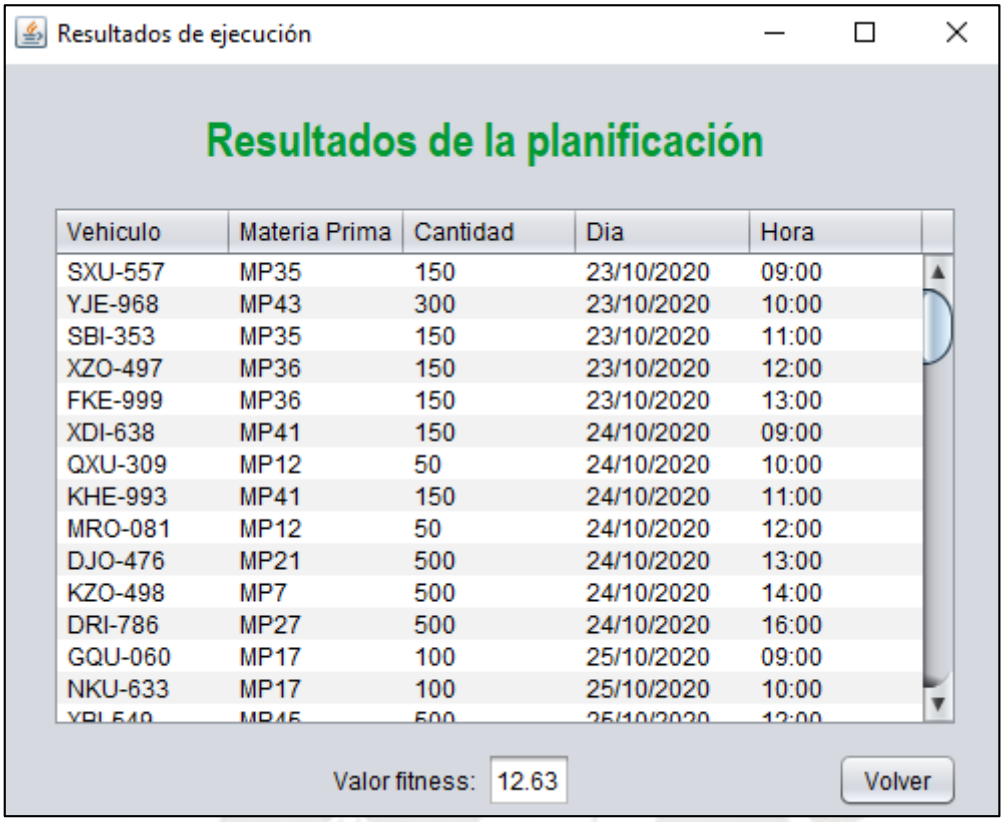


Figura 43. Pantalla de visualización de resultados de un único algoritmo. Fuente: Elaboración propia.

En caso se hayan seleccionado ambos algoritmos, se muestra la planificación obtenida por cada uno de estos, como se puede observar en la Figura 44.

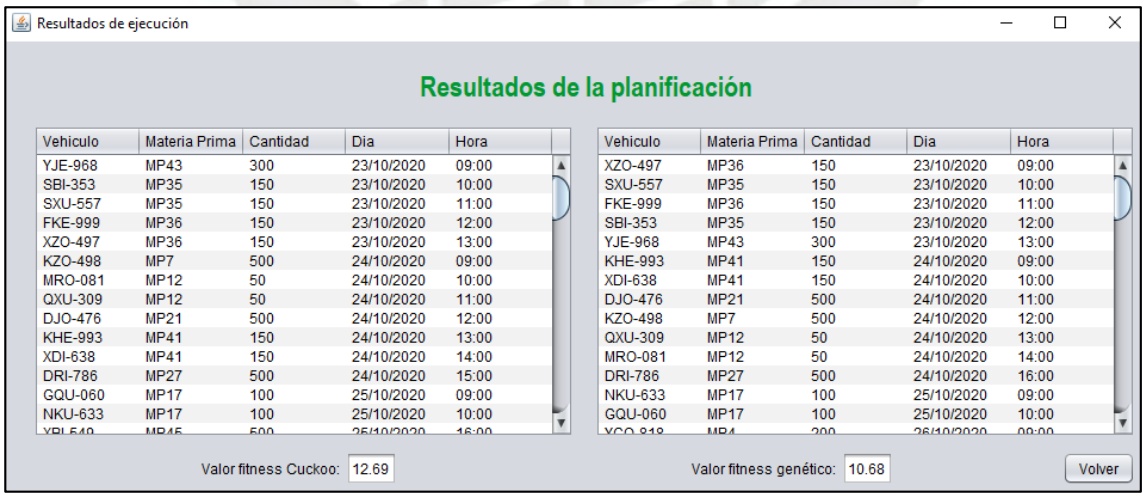


Figura 44. Pantalla de visualización de resultados de ambos algoritmos. Fuente: Elaboración propia.

10.2.2 Validaciones realizadas

Se realizaron pruebas de integración para validar que los datos utilizados para la ejecución de los algoritmos sean los capturados mediante la interfaz y la presentación de resultados esté acorde a lo entregado como mejor solución según el algoritmo utilizado. En el anexo M se presenta, a detalle, una prueba de integración para cada uno de estos.

10.3 Discusión

En este capítulo se ha alcanzado el resultado esperado R4.3, relacionado al desarrollo de la interfaz gráfica que permite la carga de datos de prueba y presentación de resultados de ejecución de los algoritmos.

El alcance de este resultado permite completar el objetivo específico 4: Desarrollar un software para la ejecución de los algoritmos Cuckoo Search y genético. La interfaz permitirá a los usuarios configurar los algoritmos y completar un flujo de planificación de citas de recepción de materia prima, obteniendo un resultado final según los datos ingresados.

Debido a la necesidad de cargar archivos para poder ejecutar alguno de los algoritmos, ya no se trabaja con datos aleatoriamente generados, si no con un conjunto predefinido de estos. Actualmente este archivo contiene pocos datos, ya que la finalidad es validar la integración entre las pantallas y los algoritmos. Más adelante, para la calibración de variables y experimentación numérica, se utilizarán conjuntos de datos de tamaño mayor.

Capítulo 11. Calibración de variables

11.1 Introducción

En este capítulo se desarrolla el resultado 5.1, relacionado a la calibración de variables para el algoritmo Cuckoo Search, con la finalidad de identificar los valores de los parámetros que maximizan el rendimiento del algoritmo y permiten encontrar soluciones de mayor calidad para el problema.

11.2 Resultados alcanzados

En este apartado se presenta el detalle de lo realizado para alcanzar el resultado esperado R5.1: "Calibración de variables".

11.2.1 Calibración de variables

Para la calibración se consideraron los principales parámetros que utiliza el algoritmo Cuckoo Search: porcentaje de descubrimiento de peores nidos, tamaño de población y número máximo de iteraciones. Para el tamaño de la población y el porcentaje mencionado, se toma como punto de partida los valores recomendados por el autor del algoritmo Cuckoo Search, para la mayoría de problemas de optimización, en su obra "Algoritmos metaheurísticos bio-inspirados", los cuales se establecen en 15 y 0.25 respectivamente (Yang, 2010).

Para cada modificación de valor en los parámetros a calibrar se realizaron 100 ejecuciones, tomando el mejor valor fitness obtenido como dato representativo y el fitness promedio del conjunto como se muestra en las siguientes secciones.

11.2.1.1 Porcentaje de descubrimiento de peores nidos (Pa)

Como se mencionó, para este parámetro se inició la calibración a partir de 0.25. Luego, se modificó este dato tanto hacia valores superiores como inferiores obteniendo los resultados presentados en la Tabla 18.

Tabla 18. Calibración del porcentaje de descubrimiento de peores nidos.

Pa	Mejor fitness obtenido	Promedio de fitness
0.15	12.6	12.98
0.20	12.55	12.93
0.25	12.6	12.90
0.3	12.45	12.86
0.35	12.53	12.86

Fuente: Elaboración propia.

A pesar de que se observa un empate en el valor fitness promedio para los dos últimos valores, el mejor caso obtenido ocurrió con el parámetro establecido en 0.3; por lo tanto, se seleccionó este para el porcentaje de descubrimiento de los peores nidos.

11.2.1.2 Tamaño de población

Para el tamaño de población o número de nidos, como se mencionó también, se iniciaron las pruebas de calibración en 15 nidos, para posteriormente evaluar valores extremos a este y analizar los resultados obtenidos.

Tabla 19. Calibración del tamaño de población.

Tamaño de población	Mejor fitness obtenido	Promedio de fitness
5	12.57	13.01
10	12.48	12.94
15	12.53	12.85
20	12.53	12.83
25	12.53	12.86

Fuente: Elaboración propia.

Debido a que el mejor valor fitness para los dos últimos casos es el mismo, se tomó la decisión de seleccionar el valor de 20 nidos para el tamaño de población en base al valor fitness promedio del conjunto de datos.

11.2.1.3 Número máximo de iteraciones

Este parámetro, a diferencia de los demás que empezaron desde un valor sugerido, fue calculado de forma empírica tomando como base diversas aplicaciones del algoritmo Cuckoo Search, obteniendo los resultados presentados en la Tabla 20.

Tabla 20. Calibración del número de iteraciones

Número de iteraciones	Mejor fitness obtenido	Promedio de fitness
50	12.6	12.85
100	12.59	12.78
250	12.5	12.72
500	12.43	12.67
1000	12.41	12.59
1500	12.35	12.59
2000	12.35	12.59

Fuente: Elaboración propia.

Como se puede observar, en los dos últimos casos tanto el mejor caso obtenido como el valor promedio de fitness se mantienen, por lo que se establece el número de iteraciones en 1500 para el algoritmo Cuckoo Search.

11.2.2 Validaciones realizadas

Este capítulo ha sido revisado por el especialista en algoritmia, quien ha dado su aprobación respecto a la información presentada en el mismo. En el anexo O se adjunta la evidencia.

11.3 Discusión

En este capítulo se ha alcanzado el resultado esperado R5.1, relacionado a la calibración de variables para el algoritmo Cuckoo Search, las cuales incluyen el porcentaje de descubrimiento de peores nidos, tamaño de población (número de nidos) y número máximo de iteraciones.

El alcance de este resultado permite definir los valores más adecuados para maximizar la calidad de las soluciones generadas por el algoritmo para el problema que se desea resolver, para lo cual fue necesario realizar modificaciones a los valores obtenidos de la literatura, con la finalidad de obtener el menor valor fitness posible.

Respecto a investigaciones anteriores y casos de estudio seleccionados en el Estado del Arte, los parámetros utilizados para la ejecución del algoritmo varían según el caso. Existen numerosas investigaciones respecto a cuáles serían los valores más adecuados para cada uno de estos; sin embargo, también se afirma que es importante analizarlos particularmente en cada aplicación según la respuesta que se desea obtener; es decir, dependen del problema.

En este proyecto se realizó la calibración de variables solo para el algoritmo Cuckoo Search debido a que es el que se desea optimizar. El algoritmo genético, por su parte, será ejecutado con valores promedio para cada parámetro, los cuales han sido establecidos según la revisión de aplicaciones del mismo.

Capítulo 12. Experimentación numérica

12.1 Introducción

En este capítulo se desarrolla el resultado 5.2, relacionado a la experimentación numérica, la cual permite determinar qué algoritmo se considera una mejor solución para el problema de planificación de citas de recepción de materia prima en empresas de producción multiplanta, tomando como métrica el valor de la función objetivo.

12.2 Resultados alcanzados

En este apartado se presenta el detalle de lo realizado para alcanzar el resultado esperado R5.2: "Informe de experimentación numérica".

12.2.1 Informe de experimentación numérica

El contenido de esta sección se divide en la recolección de datos y las pruebas de Kolmogorov-Smirnov, F de Fisher y Prueba Z, las cuales permitirán analizar las muestras de ejecución obtenidas para los algoritmos desarrollados.

12.2.1.1 Recolección de datos

Para la recolección de datos se estableció el tiempo de planificación en quince días, cada uno de ocho horas de trabajo. Luego, se realizó la planificación de 50 recepciones en este espacio de tiempo, variando la cantidad solicitada por cada materia prima y las fechas de compra, importación y límite de necesidad, ejecutando, para cada conjunto de datos, 10 repeticiones, con la finalidad de tomar el menor valor fitness obtenido como dato representativo.

Los resultados obtenidos se presentan en la Figura 45 y se detallan en el anexo P.

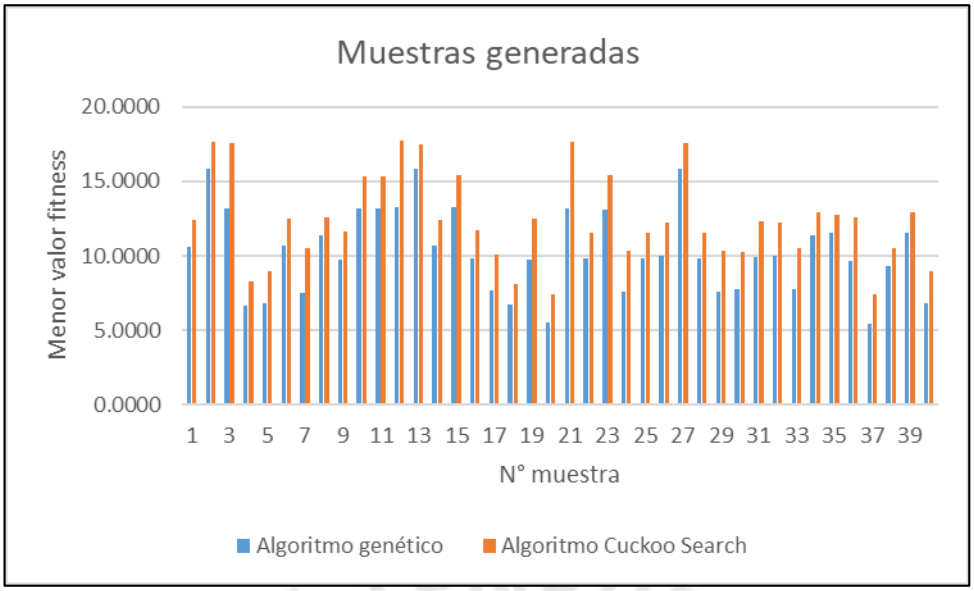


Figura 45. Menor valor fitness en la recolección de datos. Fuente: Elaboración propia.

12.2.1.2 Prueba de Kolmogorov-Smirnov

Se utilizó la prueba de Kolmogorov-Smirnov para verificar si las muestras obtenidas siguen una distribución normal. Esta prueba es necesaria ya que la normalidad en los datos es un requisito previo para la Prueba Z.

Para la prueba de Kolmogorov-Smirnov se realiza la validación de las muestras para cada algoritmo por separado, pero la hipótesis se mantiene dado que se tiene el mismo objetivo en ambos casos:

H_0: La muestra sigue una distribución normal

H_1: La muestra no sigue una distribución normal

Para el algoritmo genético:

```
> ks.test(Dataset$Algoritmo.genético,pnorm,mean(Dataset$Algoritmo.genético),sd(Dataset
$Algoritmo.genético))

        One-sample Kolmogorov-Smirnov test

data:  Dataset$Algoritmo.genético
D = 0.11068, p-value = 0.7112
alternative hypothesis: two-sided
```

Figura 46. Prueba Kolmogorov-Smirnov para el algoritmo genético. Fuente: Elaboración propia.

En la Figura 46, a un nivel de significancia de 0.05, no hay evidencia suficiente para rechazar la hipótesis nula, ya que el p-valor obtenido es de 0.7112. Por lo tanto, se concluye que la muestra sigue una distribución normal.

Para el algoritmo Cuckoo Search:

```
> ks.test(Dataset$Algoritmo.Cuckoo.Search,pnorm,mean(Dataset$Algoritmo.Cuckoo.Search),s
d(Dataset$Algoritmo.Cuckoo.Search))

        One-sample Kolmogorov-Smirnov test

data:  Dataset$Algoritmo.Cuckoo.Search
D = 0.18198, p-value = 0.1414
alternative hypothesis: two-sided
```

Figura 47. Prueba Kolmogorov-Smirnov para el algoritmo Cuckoo Search. Fuente: Elaboración propia.

En la Figura 47, a un nivel de significancia de 0.05, no hay evidencia suficiente para rechazar la hipótesis nula, ya que el p-valor obtenido es de 0.1414. Por lo tanto, se concluye que la muestra sigue una distribución normal.

12.2.1.3 Prueba F de Fisher

Otro requisito para poder realizar la Prueba Z es verificar si la varianza entre las muestras es significativamente diferente u homogénea. Para dar respuesta a este planteamiento se utiliza la Prueba F de Fisher, en la cual las hipótesis son las siguientes:

H_0: Las varianzas entre las muestras son homogéneas

H_1: Las varianzas entre las muestras son diferentes

```
> var.test(Dataset$Algoritmo.Cuckoo.Search,Dataset$Algoritmo.genético, alternative="tw
o.sided")

        F test to compare two variances

data:  Dataset$Algoritmo.Cuckoo.Search and Dataset$Algoritmo.genético
F = 1.1402, num df = 39, denom df = 39, p-value = 0.684
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.6030632 2.1558412
sample estimates:
ratio of variances
          1.140223
```

Figura 48. Prueba F de Fisher para los algoritmos. Fuente: Elaboración propia.

En la Figura 48, a un nivel de significancia de 0.05, no hay evidencia suficiente para rechazar la hipótesis nula, ya que el p-valor obtenido es de 0.684. Por lo tanto, se concluye que la varianza entre ambas muestras es homogénea.

12.2.1.4 Prueba Z

Finalmente se procede a realizar la Prueba Z, luego de garantizar la normalidad y las varianzas homogéneas entre las muestras.

En este caso las hipótesis son las siguientes:

H_0: La media del algoritmo Cuckoo Search es igual a la del algoritmo genético

H_1: La media del algoritmo Cuckoo Search es diferente a la del algoritmo genético

```
> z.test(Dataset$Algoritmo.Cuckoo.Search, Dataset$Algoritmo.genético, alternative="tw
o.sided",mu=0,sigma.x=sd(Dataset$Algoritmo.Cuckoo.Search),sigma.y=sd(Dataset$Algoritm
o.genético))

        Two-sample z-Test

data:  Dataset$Algoritmo.Cuckoo.Search and Dataset$Algoritmo.genético
z = 3.434, p-value = 0.0005947
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 0.9447418 3.4570638
sample estimates:
mean of x mean of y
 12.43194  10.23104
```

Figura 49. Prueba Z para los algoritmos. Fuente: Elaboración propia.

En la Figura 49, a un nivel de significancia de 0.05, hay evidencia suficiente para rechazar la hipótesis nula, ya que el p-valor obtenido es de 0.0005947. Por lo tanto, se concluye que la media del algoritmo Cuckoo Search es diferente a la del algoritmo genético.

Además, como se muestra en el detalle de resultados, la media del algoritmo Cuckoo Search es de 12.43194, mientras que para el algoritmo genético es de 10.23104, por lo que se puede afirmar que este último brinda soluciones de calidad superior al primero.

12.2.2 Validaciones realizadas

Este capítulo ha sido revisado por el especialista en algoritmia, quien ha dado su aprobación respecto a la información presentada en el mismo. En el anexo Q se adjunta la evidencia.

12.3 Discusión

En este capítulo se ha alcanzado el resultado esperado R5.2, relacionado al informe de experimentación numérica, el cual permite analizar estadísticamente la respuesta de los algoritmos frente al problema estudiado, tomando como métrica de evaluación el valor de la función objetivo.

El alcance de este resultado permite identificar el algoritmo que representa una mejor herramienta de solución según el análisis de las muestras de ejecución obtenidas. Así, para este caso, el algoritmo genético mostró una ligera ventaja respecto al algoritmo Cuckoo Search, según la Prueba Z.

Respecto a investigaciones anteriores y casos de estudio seleccionados en el Estado del Arte, se manifiesta expresamente la ventaja del algoritmo Cuckoo Search respecto al algoritmo genético en diversos problemas de planificación. Sin embargo, para este caso específico, esta tendencia no se ha mantenido. Los factores a los que se debe pueden ser diversos; por ejemplo, la generación aleatoria de población inicial para el algoritmo Cuckoo Search, en vez del uso de una heurística para la construcción de la misma. También influye la forma que utiliza cada uno

de los algoritmos para mejorar las posibles soluciones, siendo el casamiento y la mutación del algoritmo genético superiores al movimiento aleatorio que realiza el algoritmo Cuckoo Search, el cual, en la mayoría de los casos, no consigue ajustar mejor las planificaciones según la función objetivo diseñada.

Capítulo 13. Conclusiones y trabajos futuros

13.1 Conclusiones

En este proyecto se desarrolló una alternativa de solución para el problema de la planificación de citas de recepción de materia prima en empresas de producción multiplanta, utilizando dos algoritmos metaheurísticos, los cuales fueron diseñados e implementados para proporcionar soluciones válidas y de calidad para el mismo.

En el primer objetivo específico, se elaboraron las restricciones del problema (R1.1) y la formulación de la función objetivo que permite evaluar las posibles soluciones (R1.2). Un análisis a detalle del problema y del contexto en el que se presenta este caso de estudio fueron necesarios para plantear estas definiciones.

Para el segundo objetivo específico, relacionado al diseño del algoritmo genético, se elaboró la estructura principal para representar el cromosoma (R2.1), así como las estructuras de datos que dan soporte a la implementación del mismo, y el pseudocódigo del algoritmo genético adaptado al problema (R2.2). Para alcanzar este resultado fue importante el análisis de diversas aplicaciones del algoritmo genético, ya que existen numerosas técnicas para seleccionar, cruzar o mutar a los individuos que conforman las generaciones.

Para el tercer objetivo específico, se diseñó la estructura del nido para el algoritmo Cuckoo Search (R3.1), la cual se definió como la misma utilizada para el cromosoma del algoritmo genético, ya que esta estructura favorece la asignación de vehículos a espacios de tiempo diferentes. Para el diseño del pseudocódigo del algoritmo Cuckoo Search (R3.2) también se analizaron casos de estudio similares; sin embargo, para este algoritmo no existen numerosas formas de desarrollar cada etapa del mismo, por lo que el diseño a realizar fue más claro a diferencia del algoritmo genético.

En el cuarto objetivo específico, se realizó la codificación de ambos algoritmos (R4.1, R4.2), tomando como base el diseño realizado previamente de estructuras y pseudocódigo para cada uno. Se realizaron las pruebas necesarias a la ejecución de los métodos que los componen, verificando que las soluciones entregadas cumplan con las restricciones del problema y que, a través de las generaciones, estas sean optimizadas favorablemente para brindar una solución de calidad al problema. En este objetivo también se diseñó e implementó una interfaz gráfica (R4.3) para que el usuario pueda configurar y ejecutar los algoritmos.

Finalmente, en el quinto objetivo específico, se realizó la calibración de parámetros del algoritmo Cuckoo Search (R5.1), con la finalidad de maximizar el rendimiento del mismo para el problema estudiado. En la experimentación numérica realizado a continuación, (R5.2), se observó que las soluciones proporcionadas por el algoritmo genético son significativamente mejores respecto a las que ofrece el algoritmo Cuckoo Search, como evidencia la prueba Z, pues este último tiene una media más alta de valores resultantes para la función objetivo establecida.

13.2 Trabajos futuros

Las investigaciones futuras que se proponen a partir del proyecto realizado son las siguientes:

- Utilizar una técnica de construcción heurística o metaheurística, como el algoritmo GRASP, para la generación de la población inicial, de tal forma que esta no sea aleatoriamente obtenida en el algoritmo Cuckoo Search.
- Diseñar una mejora para la evolución de planificaciones del algoritmo Cuckoo Search, la cual permita explorar el espacio de búsqueda de forma más eficiente, tal que pueda competir contra las soluciones resultantes que entrega el algoritmo genético.

- Integrar el uso de los algoritmos metaheurísticos implementados a un sistema de información, tal que estos favorezcan la automatización del proceso de planificación de citas de recepción de materia prima en el área correspondiente.

Referencias

Alvarez, E. (2007). Multi-plant production scheduling in SMEs. *Robotics and Computer-Integrated Manufacturing*, *23*(6), 608–613.

Anca, V. (2019). Logistics and supply chain management: An overview. *Studies in Business and Economics*, *14*(2), 209–215.

Armstrong, R. A., Eperjesi, F., & Gilmartin, B. (2002). The application of analysis of variance (ANOVA) to different experimental designs in optometry. Ophthalmic and Physiological Optics, 22(3), 248–256.

Banaeianjahromi, N., Kähkönen, T., Alanne, A., & Smolander, K. (2016). Integration obstacles during ERP development. *2016 49th Hawaii International Conference on System Sciences (HICSS)*, 4697–4706.

Blum, C., & Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys* (CSUR), 35(3), 268–308.

Chen, I. (2001). Planning for ERP systems: analysis and future trend. *Business Process Management Journal* [Figura].

Deemer, P., Benefield, G., Larman, C., & Vodde, B. (2009). Información básica de SCRUM. California: Scrum Training Institute.

de Man, J.., & Strandhagen, J. (2018). Spreadsheet application still dominates enterprise resource planning and advanced planning systems. *IFAC-PapersOnLine*, *51*(11), 1224–1229.

Emmett, S. (2005). *Excellence in warehouse management: how to minimise costs and maximise value*. John Wiley & Sons.

García, L. (2016). *GESTION LOGISTICA INTEGRAL: las mejores prácticas en la cadena de abastecimiento*. Ecoe Ediciones.

Gargeya, V. & Brady, C. (2005). Success and failure factors of adopting SAP in ERP system implementation. *Business Process Management Journal*.

Git. (s.f) Control rápido de versiones. Recuperado el 18 de junio de 2020, a partir de https://git-scm.com/about

Gómez, J. (2012). Análisis y diseño de algoritmos.

Gutiérrez, Ó. (2009). Un enfoque multicriterio para la toma de decisiones en la gestión de inventarios. *Cuadernos de Administración*, *22*(38), 169–187.

Guturu, P., & Dantu, R. (2008). An impatient evolutionary algorithm with probabilistic tabu search for unified solution of some NP-hard problems in graph and set theory via clique finding. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *38*(3), 645–666.

Kemper, B., Klaassen, C., & Mandjes, M. (2014). Optimized appointment scheduling. *European Journal of Operational Research*, *239*(1), 243–255.

Kitchenham, B. (2004). Procedures for performing systematic reviews. *Keele, UK, Keele University*, *33*(2004), 1-26.

López, J. (2017). *Metaheurísticas para el análisis de datos masivos en el ámbito del transporte por carretera*. Universidad de Alcalá.

Majeed, P. G., & Kumar, S. (2014). Genetic algorithms in intrusion detection systems: A survey. *International Journal of Innovation and Applied Studies*, 5(3), 233 [Figura].

Malindzakova, M., & Zimon, D. (2019). Design Supply Cycle for Inventory Management. *TEM Journal*, *8*(3), 894.

Mecalux (2019) Limpieza industrial de almacenes: ¿cómo mantener el orden? [Figura]. Recuperado de: https://www.mecalux.es/

Mentzer, J., DeWitt, W., Keebler, J., Min, S., Nix, N., Smith, C. & Zacharia, Z. G. (2001). Defining supply chain management. *Journal of Business Logistics*, *22*(2), 1–25.

Milenkova, M., Sokolović, V., Milovanović, V. & Milić, M. (2020). Logistics: Its role, significance and approaches. *Vojnotehnički Glasnik*, *68*(1), 79–106.

NetBeans. (s. f.). NetBeans IDE - Overview. Recuperado 18 de junio de 2020, a partir de https://netbeans.org/features/index.html

Oracle. (s.f) ¿Qué es la tecnología Java y para qué la necesito? Recuperado el 18 de junio de 2020, a partir de https://www.java.com/es/download/faq/whatis_java.xml

Pandolfi, D., Villagra, A., Leguizamón, G., Orozco, S., Rasjido, J., Varas, V., & Serón, N. (2018). Metaheurísticas aplicadas de problemas de scheduling con restricciones. *XX Workshop de Investigadores En Ciencias de La Computación (WICC 2018, Universidad Nacional Del Nordeste).*

Planet Together (s.f) Multi plant scheduling to drive profits [Figura]. Recuperado de: https://www.planettogether.com/multi-plant-scheduling-to-drive-profits

Pérez, I., Cifuentes, A., Vásquez, C., & Marcela, D. (2013). Un modelo de gestión de inventarios para una empresa de productos alimenticios. *Ingeniería Industrial*, *34*(2), 227–236.

Ruescas, A. (2016). *Aplicación de técnicas metaheurísticas para la asignación de turnos de trabajo*.

RStudio. (s.f) About RStudio. Recuperado 19 de junio del 2020, a partir de https://rstudio.com/about/

Stevens, G. (1989). Integrating the supply chain. *International Journal of Physical Distribution & Materials Management*.

Whitley, E., & Ball, J. (2002). Statistics review 6: Nonparametric methods. Critical Care, 6(6), 509.

Yang, X.-S. (2010). Nature-inspired metaheuristic algorithms. *Luniver press*.

Anexos

Anexo A: Plan de proyecto

- **Justificación**

En el sector productivo, el proceso de abastecimiento de materia prima representa para las empresas uno de los factores críticos dentro de la cadena de suministro. Este proceso es el encargado de mantener en el inventario de la misma, la disponibilidad necesaria y suficiente de materia prima, con la finalidad de garantizar la continuación de los procesos de manufactura posteriores.

En este contexto, realizar una planificación consistente de citas de recepción de materia prima resulta muy difícil, debido a todos los elementos que están involucrados en esta programación. La capacidad de almacenamiento disponible, el nivel de rotación de inventarios, la naturaleza de los mismos insumos e incluso el tráfico de transportistas en la planta, son variables importantes a considerar cuando se desea optimizar este procedimiento. Es así que, si el escenario está relacionado a empresas productoras del tipo multiplanta, la complejidad de los cálculos se ve exponencialmente incrementada, ya que estas organizaciones comparten recursos entre sus diferentes sedes, por lo que una recepción de materia prima se puede realizar en cualquiera de las plantas pertenecientes a la red, si es que la solicitante no está disponible temporalmente.

Los sistemas que, en la actualidad, buscan solucionar este tipo de problemas, no tienen los métodos o herramientas necesarias para poder evaluar los parámetros y restricciones que una planificación de esta complejidad tiene. El nivel de análisis es muy básico y lineal, generando, en la mayoría de situaciones, la necesidad de reajustes posteriores por parte de los usuarios, debido a que los resultados suelen ser ineficientes y muy poco, o nada, parametrizables.

Este proyecto de fin de carrera propone una solución completa, la cual tiene el objetivo de realizar una planificación consistente de las citas de recepción de materia prima en una empresa productora del tipo multiplanta, según la evaluación de los elementos importantes en este proceso logístico. Se busca involucrar al usuario en un nivel mínimo, ya que el módulo a desarrollar tendrá el objetivo de realizar el análisis de las posibles soluciones mediante la explotación de las capacidades de un algoritmo metaheurístico denominado Cuckoo Search. Además, como la revisión sugiere, se realizará un aporte significativo al Estado del Arte estudiado, pues no se encontraron aplicaciones del algoritmo mencionado a este problema particular.

- **Viabilidad**

Viabilidad técnica

Para el desarrollo del proyecto se utilizarán herramientas de acceso libre como el IDE Netbeans, el lenguaje de programación Java y RStudio, las cuales ya han sido previamente utilizadas en diversos proyectos de software y cursos en los que el autor ha participado durante su formación académica. Además, debido a que son herramientas utilizadas masivamente por programadores e investigadores, se cuenta con numerosa documentación de soporte para resolver dudas o adquirir nuevos conocimientos sobre las mismas.

Respecto a la implementación de los algoritmos, se cuenta con el apoyo de un especialista en el área, quien posee una gran experiencia y conocimientos relacionados a algoritmos metaheurísticos. Además, mediante la revisión del Estado del Arte se identificaron casos de estudio donde se aplica el mismo algoritmo que se propone como solución en este proyecto.

Por las condiciones presentadas, no se espera tener obstáculos bloqueantes que puedan perjudicar el desarrollo del proyecto.

Viabilidad temporal

El horizonte de planificación del proyecto se estima en 94 días. Se espera cumplir con las actividades dentro de los plazos estimados, las cuales permitirán alcanzar los objetivos específicos definidos, así como los resultados esperados que los componen.

Viabilidad económica

No se requiere mayor inversión para el desarrollo del proyecto, pues todas las herramientas de software que se utilizan son gratuitas y el autor ya cuenta con la instalación exitosa de las mismas en la computadora de trabajo principal.

Viabilidad de datos

Los datos de entrada y prueba que se utilizan durante el proyecto serán proporcionados por el especialista que acompaña las actividades del mismo, por lo que no hay obstáculos respecto a los datos requeridos.

- **Alcance**

El presente proyecto pertenece al área de ciencias de la computación, específicamente relacionado a algoritmia, y tiene como objetivo desarrollar un algoritmo metaheurístico denominado Cuckoo Search para solucionar el problema de planificación de citas de recepción de materia prima en empresas productoras multiplanta.

En primer lugar, se definen los parámetros y restricciones, así como la función objetivo mediante la cual se relacionan y que es la encargada de realizar la evaluación de las posibles soluciones. Para el alcance de este proyecto se ha decidido considerar la capacidad de almacenamiento, la prioridad de recepción entre materias primas, el rango de fechas esperado y la planta de destino de la recepción, esta última debido a que se prefiere realizar la recepción en la misma planta que en alguna otra de la red, evitando costos de transporte adicionales. Otros parámetros que pueden resolver variantes del problema seleccionado no se consideran dentro del alcance.

Mediante la revisión del Estado del Arte estudiado, se ha seleccionado al algoritmo genético como el más representativo para resolver este tipo de problemas, por lo que se utilizará como apoyo y elemento de comparación durante el proceso de desarrollo del proyecto. Por lo tanto, se procede, a continuación, a diseñar los componentes bio-inspirados que forman parte de cada uno de los algoritmos. Es decir, mediante la abstracción, se desea conseguir su representación computacional, de modo que estas estructuras puedan ser utilizadas en los mismos. Se implementará un módulo que contiene un algoritmo genético y el mencionado Cuckoo Search, ambos adaptados al problema que se desea resolver.

La interacción del usuario con estos algoritmos será a través de una interfaz gráfica que permitirá realizar el ingreso de datos de entrada, así como visualizar los resultados obtenidos una vez terminada su ejecución. El alcance del proyecto no tiene por objetivo desarrollar un sistema de información, por lo que esta interfaz debe entenderse solamente como un complemento de la solución que facilita su uso.

Finalmente, se realizará una comparación entre los resultados de los algoritmos mediante experimentación numérica, con la finalidad de analizar la eficiencia de respuesta del algoritmo propuesto frente al genético.

- **Restricciones**

El presente proyecto tiene una restricción respecto a la disponibilidad del especialista de logística externo, quien se encarga de validar el documento de definición de parámetros, restricciones y función objetivo.

También se considera limitante a los recursos de hardware utilizados para ejecutar el módulo que contiene a los algoritmos metaheurísticos del proyecto, ya que una infraestructura con capacidad insuficiente para soportar esta ejecución retrasará considerablemente los tiempos esperados de respuesta.

- **Identificación de los riesgos del proyecto**

Tabla A1. Identificación de riesgos del proyecto.

Descripción	Síntomas	P	I	S	Mitigación	Contingencia
Especialista en Logística no se encuentra disponible para poder validar las variables del problema.	El especialista no tiene disponibilidad de tiempo para atender el proyecto, por diversos factores personales o profesionales.	0.4	0.5	0.2	Planificar adecuadamente la participación del especialista en logística, con la finalidad de que se conozca con anticipación los días en que se necesitará su apoyo.	El asesor del tema de tesis asume el rol de especialista en logística, ya que tiene los conocimientos necesarios.

Fuente: Elaboración propia.

Donde:

- P: Probabilidad
- I: Impacto
- S: Severidad (PxI)

- **Estructura de descomposición del trabajo (EDT)**

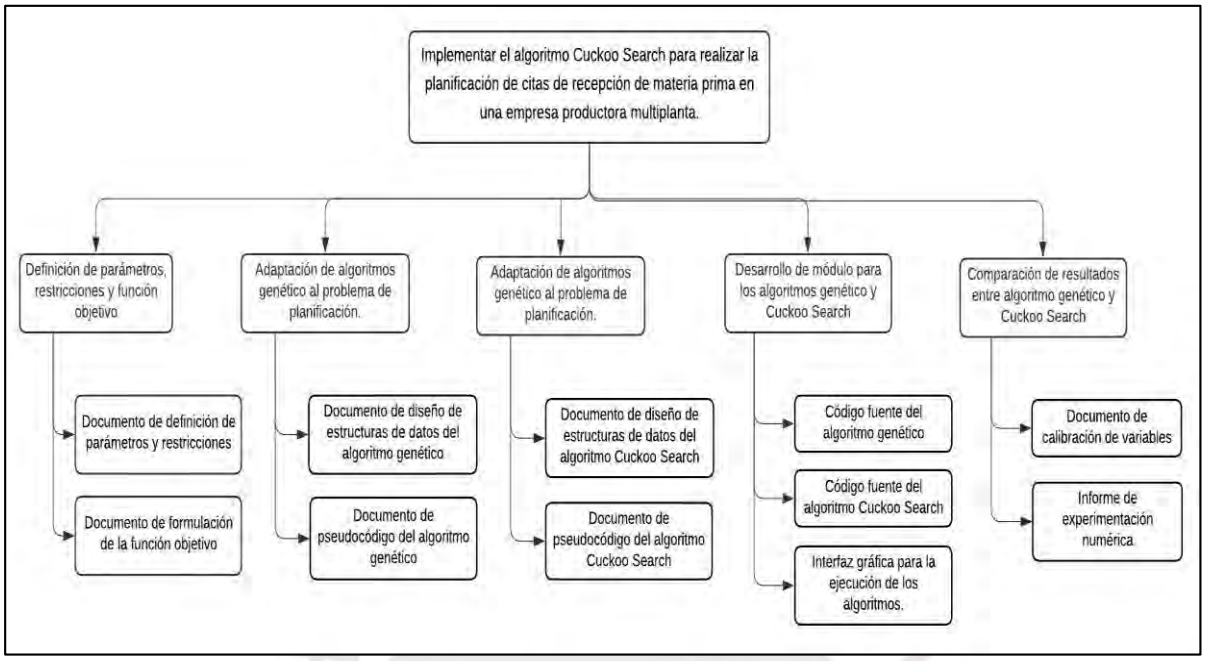


Figura A1. Estructura de descomposición del trabajo. Fuente: Elaboración Propia.

- **Lista de tareas**

Tabla A2. Lista de tareas del proyecto.

Tarea	Duración estimada (días)	Esfuerzo asociado(horas-personas)	Costo estimado
Objetivo 1: Definir los parámetros, restricciones y cómo se relacionan mediante la función objetivo, la cual sirve para realizar la evaluación de posibles soluciones.			
R1.1 Lista de definición de parámetros y restricciones para el problema de planificación.			
Elaboración del documento con la lista de parámetros y restricciones.	7	21	420
Reunión con el asesor.	1	1	50
Validación del documento por parte del especialista en logística.	1	2	100
R1.2 Formulación de la función objetivo para el problema de planificación.			
Elaboración del documento con la formulación de la función objetivo.	7	21	420

Reunión con el asesor.	1	1	50
Validación del documento por parte del especialista en algoritmia.	1	2	100
Objetivo 2: Adaptar el algoritmo genético para resolver el problema de planificación.			
R2.1 Diseño de estructuras de datos que sirven de soporte para el algoritmo genético adaptado al problema de planificación.			
Elaboración del documento de diseño de estructura de datos que sirve de soporte para el algoritmo genético adaptado al problema de planificación.	7	21	420
Reunión con el asesor.	1	1	50
Validación del documento por parte del especialista en algoritmia.	1	2	100
R2.2 Algoritmo genético adaptado al problema de planificación.			
Elaboración del documento que contiene el pseudocódigo del algoritmo genético adaptado al problema de planificación.	7	21	420
Reunión con el asesor.	1	1	50
Pruebas de caja blanca sobre la base de resultados conocidos.	5	15	300
Validación del diseño del algoritmo por parte del especialista en algoritmia.	2	4	200
Objetivo 3: Diseñar el algoritmo Cuckoo Search para resolver el problema de planificación.			
R3.1 Diseño de estructuras de datos que sirven de soporte para el algoritmo Cuckoo Search adaptado al problema de planificación.			
Elaboración del documento de diseño de estructura de datos que sirve de soporte para el algoritmo Cuckoo Search adaptado al problema de planificación.	7	21	420
Reunión con el asesor.	1	1	50
Validación del documento por parte del especialista en algoritmia.	1	2	100

R3.2 Algoritmo Cuckoo Search diseñado para resolver el problema de planificación.			
Elaboración del documento que contiene el pseudocódigo del algoritmo Cuckoo Search diseñado para resolver el problema de planificación.	7	21	420
Reunión con el asesor.	1	1	50
Pruebas de caja blanca sobre la base de resultados conocidos.	5	15	300
Validación del diseño del algoritmo por parte del especialista en algoritmia.	2	4	200
Objetivo 4: Desarrollar un software para la ejecución de los algoritmos Cuckoo Search y genético.			
R4.1 Codificación del algoritmo genético adaptado al problema de planificación.			
Codificación del algoritmo genético adaptado al problema de planificación.	7	21	420
Reunión con el asesor.	1	1	50
R4.2 Codificación del algoritmo Cuckoo Search diseñado para resolver el problema de planificación.			
Codificación del algoritmo Cuckoo Search diseñado para resolver el problema de planificación.	7	21	420
Reunión con el asesor.	1	1	50
R4.3 Interfaz gráfica para la ejecución de los algoritmos.			
Codificación de la interfaz gráfica.	4	12	240
Reunión con el asesor.	1	1	50
Objetivo 5: Desarrollar la comparación de resultados entre los algoritmos Cuckoo Search y genético.			
R5.1 Calibración de variables.			
Elaboración del documento de calibración de variables para los algoritmos.	10	30	600
Reunión con el asesor.	1	1	50
Validación del documento por parte del especialista en algoritmia.	2	4	200

R5.2 Informe de experimentación numérica.			
Elaboración del informe de experimentación numérica.	7	21	420
Reunión con el asesor.	1	1	50
Validación del documento por parte del especialista en algoritmia.	1	2	100

Fuente: Elaboración propia.

- **Cronograma del proyecto**

Tabla A3. Cronograma del proyecto.

Semana	**Tarea**
1	**Objetivo 1:** Definir los parámetros, restricciones y cómo se relacionan mediante la función objetivo, la cual sirve para realizar la evaluación de posibles soluciones.
	R1.1 Lista de definición de parámetros y restricciones para el problema de planificación.
	Elaboración del documento con la lista de parámetros y restricciones.
	Reunión con el asesor.
	Validación del documento por parte del especialista en logística.
	R1.2 Formulación de la función objetivo para el problema de planificación.
	Elaboración del documento con la formulación de la función objetivo.
	Reunión con el asesor.
	Validación del documento por parte del especialista en algoritmia.
Exposición 1	
	Levantamiento de observaciones de la semana 1.
	Objetivo 2: Adaptar el algoritmo genético para resolver el problema de planificación.
	R2.1 Diseño de estructuras de datos que sirven de soporte para el algoritmo genético adaptado al problema de planificación.

2	**Objetivo 3:** Diseñar el algoritmo Cuckoo Search para resolver el problema de planificación
	R3.1 Diseño de estructuras de datos que sirven de soporte para el algoritmo Cuckoo Search adaptado al problema de planificación.
	Elaboración del documento de diseño de estructuras de datos para ambos algoritmos.
	Reunión con el asesor.
	Validación del documento por parte del especialista en algoritmia.
Exposición 2	
3	Levantamiento de observaciones de la semana 2.
	Objetivo 2: Adaptar el algoritmo genético para resolver el problema de planificación.
	R2.2 Algoritmo genético adaptado al problema de planificación.
	Elaboración del documento que contiene el pseudocódigo del algoritmo genético adaptado al problema de planificación.
	Reunión con el asesor.
	Pruebas de caja blanca sobre la base de resultados conocidos.
	Validación del diseño del algoritmo por parte del especialista en algoritmia.
Exposición 3	
4	Levantamiento de observaciones de la semana 3.
	Objetivo 3: Diseñar el algoritmo Cuckoo Search para resolver el problema de planificación
	R3.2 Algoritmo Cuckoo Search diseñado para resolver el problema de planificación.
	Elaboración del documento que contiene el pseudocódigo del algoritmo Cuckoo Search diseñado para resolver el problema de planificación.
	Reunión con el asesor.
	Pruebas de caja blanca sobre la base de resultados conocidos.
	Validación del diseño del algoritmo por parte del especialista en algoritmia.
Exposición 4	
	Levantamiento de observaciones de la semana 4.

5	**Objetivo 4:** Desarrollar un software para la ejecución de los algoritmos Cuckoo Search y genético.
	R4.1 Codificación del algoritmo genético adaptado al problema de planificación.
	Codificación del algoritmo genético adaptado al problema de planificación.
	Reunión con el asesor.
	Redacción del entregable.
ENTREGABLE PARCIAL	
6	**R4.2** Codificación del algoritmo Cuckoo Search diseñado para resolver el problema de planificación.
	Codificación del algoritmo Cuckoo Search diseñado para resolver el problema de planificación.
	Reunión con el asesor.
	Redacción del entregable.
7	Levantamiento de observaciones del entregable parcial.
	R4.3 Interfaz gráfica para la ejecución de los algoritmos.
	Codificación de la interfaz gráfica.
	Reunión con el asesor.
	Redacción del entregable.
Exposición 5	
8	**Objetivo 5:** Desarrollar la comparación de resultados entre los algoritmos Cuckoo Search y genético.
	R5.1 Calibración de variables.
	Elaboración del documento de calibración de variables para los algoritmos.
	Reunión con el asesor.
	Validación del documento por parte del especialista en algoritmia.
9	**R5.2** Informe de experimentación numérica.
	Elaboración del informe de experimentación numérica.
	Reunión con el asesor.
	Validación del documento por parte del especialista en algoritmia.
Exposición 6	

10	Levantamiento de observaciones de las semana 8 y 9.
	Correcciones finales.
	Reunión con el asesor.
ENTREGABLE FINAL	

Fuente: Elaboración propia.

- **Lista de recursos**

  - **Personas involucradas y necesidades de capacitación**

Tabla A4. Personas involucradas y necesidades de capacitación.

Persona involucrada	Necesidad de capacitación
Steven Alonso Labajos Trigoso	Capacitación en diseño de algoritmos, pruebas de caja blanca y calibración de variables.
Rony Cueva Moscoso	Ninguna

Fuente: Elaboración propia.

  - **Materiales requeridos para el proyecto**

Tabla A5. Materiales requeridos para el proyecto.

Material requerido	Objetivo de uso
Útiles de escritorio	Se utilizará para diversos fines de apoyo asociados a cálculos, diseño y pruebas.

Fuente: Elaboración propia.

- **Estándares utilizados en el proyecto**

Tabla A6. Estándares utilizados en el proyecto.

Estándar utilizado	Objetivo de uso
Guía PMBOK	Se utilizará como base metodológica del proyecto, ya que se sigue una estructura jerárquica de tareas para la obtención de los entregables requeridos.
Scrum	Se utilizarán algunas buenas prácticas del marco de trabajo para una colaboración eficaz entre los involucrados y entrega de resultados.

Fuente: Elaboración propia.

- **Equipamiento requerido**

Tabla A7. Equipamiento requerido para el proyecto.

Equipamiento requerido	Objetivo de uso
Una computadora personal.	Se utilizará para la codificación de los algoritmos y las interfaces gráficas, así como para realizar las pruebas funcionales. También se utilizará para la calibración de variables y la experimentación numérica.

Fuente: Elaboración propia.

- **Herramientas requeridas**

Tabla A8. Herramientas requeridas para el proyecto.

Herramientas requeridas	Objetivo de uso
Java	Se utilizará como lenguaje de programación para el desarrollo de los algoritmos e interfaces gráficas.
NetBeans IDE	Se utilizará como el entorno de desarrollo integrado para la codificación.
Git	Se utilizará como herramienta de control de versiones para el código fuente.
RStudio	Se utilizará en el análisis de datos para el informe de experimentación numérica.

Fuente: Elaboración propia.

- **Costeo del proyecto**

Tabla A9. Costeo del proyecto.

Ítem	Descripción	Unidad	Cantidad	Valor Unidad (S/.)	Monto Parcial (S/.)	Monto Total (S/.)
0	**Costo total del proyecto**	**---**	**---**	**---**	**---**	**7375**
1.	**Estudiantes o tesistas**	**---**	**---**	**---**	**---**	**5000**
1.1	Steven Alonso Labajos Trigoso	Horas	250	20	5000	
2.	**Otros participantes**	**---**	**---**	**---**	**---**	**1600**
2.1	Especialista en logística	Horas	2	50	100	
2.2	Especialista en algoritmia	Horas	30	50	1500	
3.	**Materiales e insumos**	**---**	**---**	**---**	**---**	**25**
3.1	Útiles de escritorio	Unidad	5	5	25	
4.	**Bienes y equipos** **Unid1** **Cant1**	**Unid2**	**Cant2**	**-**	**-**	**750**

4.1	Computadoras	Equipo	1	Horas	150	5		

Fuente: Elaboración propia.

Anexo B: Formulario de extracción de datos

Archivo “20141860_StevenLabajos_RonyCueva_Formulario”, presente en el entregable.

Anexo C: Pruebas realizadas sobre la función objetivo.

$$min \sum_{n=1}^{N} \sum_{a=1}^{A} \frac{\sum_{m=0}^{M} [Ur * (LN - FR)]_{man}}{CAlm_{an}}$$

Caso de prueba N°1:

Tabla C1. Datos de entrada para el caso de prueba N°1.

Capacidad de almacenamiento disponible

Almacén	Capacidad	Unidad
Almacen 1	100	bolsas
Almacen 2	200	tn
Almacen 3	100	pallets

Solicitudes de recepción

MP	Cantidad	Unidad	LN	FC	FI
Cemento T1	70	bolsas	27-ago	21-ago	
Cemento T2	20	bolsas	27-ago	23-ago	
Arena fina	60	tn	25-ago		20-ago
Arena gruesa	100	tn	29-ago	22-ago	
Pegamento	90	pallets	30-ago	25-ago	

Días de margen de seguridad: 2

Nota: Las nomenclaturas utilizadas son las mismas que se detallaron en el Capítulo 4: materia prima (MP), fecha límite de necesidad (LN), fecha de compra (FC) y fecha de ingreso de importación (FI). Fuente: Elaboración propia.

Solución A:

Tabla C2. Solución A para el caso de prueba N°1.

Planificación resultante

FR	MP	Cantidad	Unidad	MP	Cantidad	Unidad
20-ago	Arena fina	60	tn			
21-ago						
22-ago						
23-ago	Cemento T2	20	bolsas			
24-ago						
25-ago	Cemento T1	70	bolsas			
26-ago	Arena gruesa	50	tn	Pegamento	90	pallets
27-ago	Arena gruesa	50	tn			
28-ago						
29-ago						
30-ago						

Disponibilidad de almacenamiento por día

	20-ago	21-ago	22-ago	23-ago	24-ago	25-ago	26-ago	27-ago	28-ago	29-ago	30-ago
Almacen 1	100	100	100	100	80	80	10	10	10	10	10
Almacen 2	200	140	140	140	140	140	140	90	40	40	40
Almacen 3	100	100	100	100	100	100	100	10	10	10	10

FUNCIÓN OBJETIVO: 9.832539683

Fuente: Elaboración propia.

Solución B:

Tabla C3. Solución B para el caso de prueba N°1.

Planificación resultante

FR	MP	Cantidad	Unidad	MP	Cantidad	Unidad
20-ago	Arena fina	60	tn			
21-ago						
22-ago						
23-ago						
24-ago						
25-ago	Cemento T1	70	bolsas	Cemento T2	20	bolsas
26-ago	Arena gruesa	50	tn			
27-ago	Arena gruesa	50	tn	Pegamento	90	pallets
28-ago						
29-ago						
30-ago						

Disponibilidad de almacenamiento por día

	20-ago	21-ago	22-ago	23-ago	24-ago	25-ago	26-ago	27-ago	28-ago	29-ago	30-ago
Almacen 1	100	100	100	100	100	100	10	10	10	10	10
Almacen 2	200	140	140	140	140	140	140	90	40	40	40
Almacen 3	100	100	100	100	100	100	100	100	10	10	10

FUNCIÓN OBJETIVO: 8.182539683

Nota: El valor de la función objetivo mejora respecto a la anterior solución, ya que el Cemento T2 se está recepcionando en una fecha más cercana a su fecha límite de necesidad. Fuente: Elaboración propia.

Solución C:

Tabla C4. Solución C para el caso de prueba N°1.

Planificación resultante

FR	MP	Cantidad	Unidad	MP	Cantidad	Unidad
20-ago	Arena fina	60	tn			
21-ago						
22-ago						
23-ago						
24-ago						
25-ago	Cemento T1	70	bolsas	Cemento T2	20	bolsas
26-ago						
27-ago	Arena gruesa	100	tn			
28-ago	Pegamento	90	pallets			
29-ago						
30-ago						

Disponibilidad de almacenamiento por día

	20-ago	21-ago	22-ago	23-ago	24-ago	25-ago	26-ago	27-ago	28-ago	29-ago	30-ago
Almacen 1	100	100	100	100	100	100	10	10	10	10	10
Almacen 2	200	140	140	140	140	140	140	140	40	40	40
Almacen 3	100	100	100	100	100	100	100	100	100	10	10

FUNCIÓN OBJETIVO: 6.528571429

Nota: El valor de la función objetivo mejora respecto a la anterior solución, ya que se está recibiendo la totalidad de la arena gruesa en una sola recepción y en una fecha más cercana a la fecha límite de necesidad. Además, de igual manera, el pegamento se almacena una menor cantidad de días. Fuente: Elaboración propia.

Caso de prueba N°2:

Tabla C5. Datos de entrada para el caso de prueba N°2.

Capacidad de almacenamiento disponible

Almacén	Capacidad	Unidad
Almacen 1	150	sacos
Almacen 2	10	tn
Almacen 3	100	galones

Solicitudes de recepción

MP	Cantidad	Unidad	LN	FC	FI
Harina	50	sacos	17-sep	12-sep	
Levadura	50	sacos	16-sep	13-sep	
Azucar	2	tn	19-sep		15-sep
Leche	80	galones	20-sep	17-sep	
Polvo de hornear	30	sacos	20-sep		16-sep

Días de margen de seguridad: 1

Fuente: Elaboración propia.

Solución A:

Tabla C6. Solución A para el caso de prueba N°2.

Planificación resultante

FR	MP	Cantidad	Unidad	MP	Cantidad	Unidad
10-sep						
11-sep						
12-sep						
13-sep						
14-sep	Levadura	50	sacos			
15-sep	Azucar	2	tn			
16-sep	Polvo de hornear	30	sacos	Harina	50	sacos
17-sep						
18-sep	Leche	40	galones			
19-sep	Leche	40	galones			
20-sep						

Disponibilidad de almacenamiento por día

	10-sep	11-sep	12-sep	13-sep	14-sep	15-sep	16-sep	17-sep	18-sep	19-sep	20-sep
Almacen 1	150	150	150	150	150	100	100	20	20	20	20
Almacen 2	10	10	10	10	10	10	8	8	8	8	8
Almacen 3	100	100	100	100	100	100	100	100	100	60	20

FUNCIÓN OBJETIVO: 4.633333333

Fuente: Elaboración propia.

Solución B:

Tabla C7. Solución B para el caso de prueba N°2.

Planificación resultante

FR	MP	Cantidad	Unidad	MP	Cantidad	Unidad
10-sep						
11-sep						
12-sep						
13-sep						
14-sep	Levadura	50	sacos	Harina	50	sacos
15-sep	Azucar	2	tn			
16-sep	Polvo de hornear	30	sacos			
17-sep						
18-sep	Leche	80	galones			
19-sep						
20-sep						

Disponibilidad de almacenamiento por día

	10-sep	11-sep	12-sep	13-sep	14-sep	15-sep	16-sep	17-sep	18-sep	19-sep	20-sep
Almacen 1	150	150	150	150	150	50	50	20	20	20	20
Almacen 2	10	10	10	10	10	10	8	8	8	8	8
Almacen 3	100	100	100	100	100	100	100	100	100	20	20

FUNCIÓN OBJETIVO: 6.466666667

Nota: El valor de la función objetivo aumenta respecto a la anterior solución, ya que tanto la harina como la leche se están recepcionando en fechas más distantes respecto a su fecha límite de necesidad. Fuente: Elaboración propia.

Solución C:

Tabla C8. Solución C para el caso de prueba N°2.

Planificación resultante

FR	MP	Cantidad	Unidad	MP	Cantidad	Unidad
10-sep						
11-sep						
12-sep						
13-sep	Harina	30	sacos			
14-sep	Levadura	50	sacos	Harina	20	sacos
15-sep	Azucar	2	tn			
16-sep	Polvo de hornear	30	sacos			
17-sep	Leche	80	galones			
18-sep						
19-sep						
20-sep						

Disponibilidad de almacenamiento por día

	10-sep	11-sep	12-sep	13-sep	14-sep	15-sep	16-sep	17-sep	18-sep	19-sep	20-sep
Almacen 1	150	150	150	150	120	50	50	20	20	20	20
Almacen 2	10	10	10	10	10	10	8	8	8	8	8
Almacen 3	100	100	100	100	100	100	100	100	20	20	20

FUNCIÓN OBJETIVO: 7.733333333

Nota: El valor de la función objetivo aumenta respecto a la anterior solución, ya que la harina se está recibiendo en 2 recepciones parciales. Asimismo, la leche se recibe un día antes respecto a la solución anterior, manteniendo esta materia prima por más tiempo en el almacén. Fuente: Elaboración propia.

Anexo D: Conformidad del especialista en logística para el capítulo 4.

Lima, 04 de septiembre del 2020

ACTA DE CONFORMIDAD

Mediante el presente documento el Sr. CUEVA MOSCOSO RONY, especialista en logística, da conformidad de que el Sr. LABAJOS TRIGOSO STEVEN ALONSO, estudiante del curso de Proyecto de Tesis 2, ha presentado el documento que contiene la definición de parámetros, restricciones y formulación de la función objetivo para el tema de tesis que desarrolla con la información adecuada para alcanzar los objetivos esperados.

Ing. Rony Cueva Moscoso

DNI 09942265

Anexo E: Diseño de estructuras de datos auxiliares para la implementación de los algoritmos del proyecto.

- **Almacén**: Tiene la información respecto al tipo de materia prima que almacena y la capacidad de almacenamiento disponible al inicio de la planificación.

Unidad de materia prima	Capacidad inicial
Sacos	500

- **Gestor de almacenes**: Agrupa los almacenes y maneja la matriz de capacidades para los mismos.

Lista de almacenes	
{A1, A2, A3 … An}	Matriz de capacidades

- **Matriz de capacidades**: Mantiene información actualizada respecto a la capacidad disponible de los almacenes para cada día.

	Almacén 1	Almacén 2	…	Almacén n
Día 1	100	80		20
…				
Día n	400	400		300

- **Recepción**: Tiene información respecto a las características principales de las solicitudes de materia prima.

Materia prima	Cantidad	Unidad	Fecha de compra	Fecha de importación	Fecha límite
Leche	50	galones	08/09/2020	-	20/09/2020

- **Gestor de recepciones**: Agrupa las recepciones y maneja la matriz de seguimiento para las mismas.

Lista de recepciones	
{R1, R2, R3 … Rn}	Matriz de seguimiento

- **Matriz de seguimiento**: Mantiene información actualizada respecto a la cantidad recibida de cada materia prima, con la finalidad de garantizar la recepción de la totalidad de las necesidades.

	Recepción 1	Recepción 2	…	Recepción n
Solicitado	100	80		40
Actual	60	80		30

- **Vehículo**: Tiene información respecto a la cantidad de materia prima que transporta, así como a cuál está asociado.

Recepción asociada	Cantidad transportada
R1	30

- **Gestor de vehículos**: Agrupa los vehículos para facilitar al acceso a la información.

Lista de vehículos
{V1, V2, V3 … Vn}

Anexo F: Conformidad del especialista en algoritmia para el capítulo 5.

Lima, 10 de septiembre del 2020

ACTA DE CONFORMIDAD

Mediante el presente documento el Sr. CUEVA MOSCOSO RONY, especialista en algoritmia, da conformidad de que el Sr. LABAJOS TRIGOSO STEVEN ALONSO, estudiante del curso de Proyecto de Tesis 2, ha presentado el documento que contiene el diseño de las estructuras de datos para la implementación de ambos algoritmos trabajados en el proyecto. La información es adecuada para alcanzar los objetivos esperados.

Ing. Rony Cueva Moscoso

DNI 09942265

Anexo G: Pruebas de caja blanca al pseudocódigo del algoritmo genético.

Tabla G1. Datos iniciales para prueba de caja blanca del algoritmo genético.

Datos del algoritmo

Maxima generacion	3
Tamaño de población	2
Conservación de cromosomas	0.2
Tasa de casamiento	0.5
Tasa de mutación	0.1

Capacidad inicial de almacenamiento

Almacén	Capacidad	Unidad
Almacen 1	150	sacos
Almacen 2	10	tn
Almacen 3	100	galones

Recepciones solicitadas

MP	Cantidad	Unidad	LN	FC	FI	Vehículo
Harina	50	sacos	Día 3	Día 1		V01
Levadura	50	sacos	Día 3	Día 1		V02
Azucar	2	tn	Día 1		Día 1	V03
Leche	80	galones	Día 2	Día 1		V04
Polvo de hornear	30	sacos	Día 2		Día 1	V05

Días de margen de seguridad: 0

Nota: Las nomenclaturas utilizadas son las mismas que se detallaron en el Capítulo 4: materia prima (MP), fecha límite de necesidad (LN), fecha de compra (FC) y fecha de ingreso de importación (FI). La fecha límite de necesidad (LN) ya considera los días de márgenes de seguridad. Fuente: Elaboración propia.

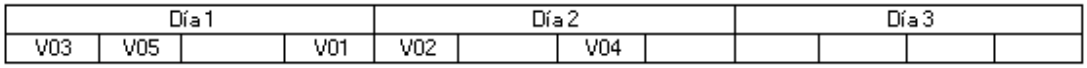
Tabla G2. Construcción de la población inicial.

Solución esperada

Día 1				Día 2				Día 3			
V03		V05			V04				V02	V01	

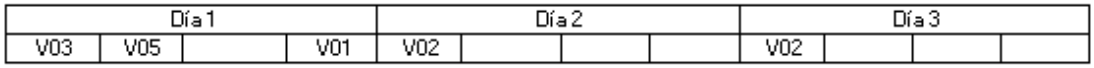
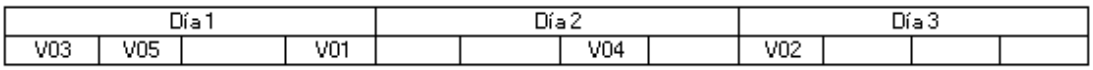
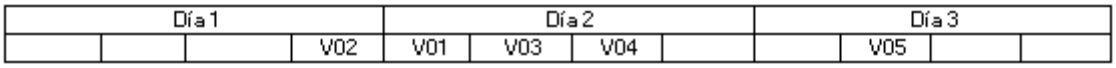
Soluciones aleatorias obtenidas

Día 1				Día 2				Día 3			
V03	V05	V04	V01					V02			

Función objetivo: 1.667 (Cromosoma 1)

Función objetivo: 1.581 (Cromosoma 2)

Nota: En esta tabla se presenta cuál sería una solución esperada como resultante del algoritmo. También, se generan aleatoriamente dos posibles soluciones válidas al problema, las cuales conforman la población inicial. Fuente: Elaboración propia.

Tabla G3. Resultados de la primera generación.

Generación 1
Hijos del casamiento entre cromosomas 1 y 2

No válido (V02 repetido)

Función objetivo: 0.8667 (Cromosoma 3)
Mutación Cromosoma 2

No válido (Restricción de importaciones)
Depuración
Cromosoma 3 por conservación, cromosoma 2 por ruleta

Nota: En esta tabla se presentan los resultados de la primera generación: el método de casamiento, mutación y la depuración de la población para retornar a su cantidad original. Fuente: Elaboración propia.

Tabla G4. Resultados de la segunda generación.

Generación 2												
Hijos del casamiento entre cromosomas 3 y 2												
Día 1				Día 2				Día 3				
V03	V05		V01			V04						
No válido (No están todos los vehículos)												
Día 1				Día 2				Día 3				
V03	V05		V01	V02		V04		V02				
No válido (V02 repetido)												
Mutación Cromosoma 3												
Día 1				Día 2				Día 3				
		V03	V05			V04		V02		V01		
Función objetivo: 0.2 (Cromosoma 4)												
Depuración												
Cromosoma 4 por conservación, cromosoma 2 por ruleta												

Nota: En esta tabla se presentan los resultados de la segunda generación: el método de casamiento, mutación y la depuración de la población para retornar a su cantidad original. Fuente: Elaboración propia.

Tabla G5. Resultados de la tercera generación.

Generación 3												
Hijos del casamiento entre cromosomas 4 y 2												
Día 1				Día 2				Día 3				
V03	V05	V03	V05			V04		V02		V01		
No válido (V03 repetido)												
Día 1				Día 2				Día 3				
V03	V05		V01	V02		V04				V01		
No válido (V01 repetido)												

<table>
<tr><td>Mutación Cromosoma 4</td></tr>
<tr><td>
<table>
<tr><td colspan="4">Día 1</td><td colspan="4">Día 2</td><td colspan="4">Día 3</td></tr>
<tr><td>V03</td><td>V05</td><td></td><td>V01</td><td>V02</td><td></td><td>V04</td><td></td><td></td><td>V03</td><td></td><td></td></tr>
</table>
</td></tr>
<tr><td>No válido (V03 repetido)</td></tr>
<tr><td>Depuración</td></tr>
<tr><td>Cromosoma 4 por conservación, cromosoma 2 por ruleta</td></tr>
<tr><td>Mejor solución resultante</td></tr>
<tr><td>Función objetivo: 0.2 (Cromosoma 4)</td></tr>
</table>

Nota: En esta tabla se presentan los resultados de la tercera generación: el método de casamiento, mutación y la depuración de la población para retornar a su cantidad original. Además, dado que esta es la última generación, se muestra cuál sería la mejor solución resultante de la ejecución del algoritmo. Fuente: Elaboración propia.

Anexo H: Conformidad del especialista en algoritmia para el capítulo 6.

Lima, 18 de septiembre del 2020

ACTA DE CONFORMIDAD

Mediante el presente documento el Sr. CUEVA MOSCOSO RONY, especialista en algoritmia, da conformidad de que el Sr. LABAJOS TRIGOSO STEVEN ALONSO, estudiante del curso de Proyecto de Tesis 2, ha presentado el documento que contiene el diseño del pseudocódigo del algoritmo genético adaptado al problema de planificación. Se han revisado los operadores de selección, casamiento y mutación, así como el proceso de depuración final de la población. La información es adecuada para alcanzar los objetivos esperados.

Ing. Rony Cueva Moscoso

DNI 09942265

Anexo I: Pruebas de caja blanca al pseudocódigo del algoritmo Cuckoo Search.

Tabla I1. Datos iniciales para la prueba de caja blanca del algoritmo Cuckoo Search

Datos del algoritmo

Días de margen de seguridad	0
Maxima generacion	3
Número de nidos	2
Porcentaje de nidos descubiertos	0.25

Capacidad inicial de almacenamiento

Almacén	Capacidad	Unidad
Almacen 1	150	sacos
Almacen 2	10	tn
Almacen 3	100	galones

Recepciones solicitadas

MP	Cantidad	Unidad	LN	FC	FI	Vehículo
Harina	50	sacos	Día 3	Día 1		V01
Levadura	50	sacos	Día 3	Día 1		V02
Azucar	2	tn	Día 1		Día 1	V03
Leche	80	galones	Día 2	Día 1		V04
Polvo de hornear	30	sacos	Día 2		Día 1	V05

Nota: Las nomenclaturas utilizadas son las mismas que se detallaron en el Capítulo 4: materia prima (MP), fecha límite de necesidad (LN), fecha de compra (FC) y fecha de ingreso de importación (FI). La fecha límite de necesidad (LN) ya considera los días de márgenes de seguridad. Fuente: Elaboración propia.

Tabla I2. Construcción de la población inicial.

Solución esperada

Día 1				Día 2				Día 3			
V03		V05			V04				V02	V01	

Soluciones aleatorias obtenidas

Día 1				Día 2				Día 3			
V03	V05	V04	V01					V02			

Función objetivo: 1.667 (Nido 1)

Día 1				Día 2				Día 3			
V03	V05		V01	V02		V04					

Función objetivo: 1.581 (Nido 2)

Nota: En esta tabla se presenta cuál sería una solución esperada como resultante del algoritmo. También, se generan aleatoriamente dos posibles soluciones válidas al problema, las cuales conforman la población inicial. Fuente: Elaboración propia.

Tabla I3. Resultados de la primera iteración.

Iteración 1

Ejemplo de cálculo de desplazamiento por Lévy para elemento del Nido 2

tamaño nido	12
u	0.6966
v	1
paso	0.6966
tamaño paso	0.6966
actual	1
nuevo	3.0898

Luego de iterar para cada elemento del Nido 2

Día 1				Día 2				Día 3			
V05	V01			V03	V04	V02					

No válido (Restricción de importaciones)

Se descubre el Nido 1 y se construye una nueva solución

Día 1				Día 2				Día 3			
V03	V05		V02				V04				V01

Función objetivo: 0.8667 (Nido 3)

Nota: En esta tabla se presentan los resultados de la primera iteración: la modificación del nido por distribución de Lévy y el descarte de nidos menos favorables para la construcción de nuevas soluciones. Fuente: Elaboración propia.

Tabla I4. Resultados de la segunda iteración.

Iteración 2											
Luego de iterar para cada elemento del Nido 3											
Día 1				Día 2				Día 3			
V03		V01		V04	V02		V05				
No válido (Restricción de importaciones)											
Se descubre el Nido 2 y se construye una nueva solución											
Día 1				Día 2				Día 3			
V03	V05	V01		V04	V02						
Función objetivo: 1.5809 (Nido 4)											

Nota: En esta tabla se presentan los resultados de la segunda iteración: la modificación del nido por distribución de Lévy y el descarte de nidos menos favorables para la construcción de nuevas soluciones. Fuente: Elaboración propia.

Tabla I5. Resultados de la tercera iteración.

Iteración 3											
Luego de iterar para cada elemento del Nido 4											
Día 1				Día 2				Día 3			
V02			V01		V05			V04		V03	
No válido (Restricción de importaciones)											
Se descubre el Nido 4 y se construye una nueva solución											
Día 1				Día 2				Día 3			
V03	V05	V04	V01					V02			
Función objetivo: 1.667 (Nido 5)											
Mejor solución resultante											
Función objetivo: 0.8667 (Nido 3)											

Nota: En esta tabla se presentan los resultados de la tercera iteración: la modificación del nido por distribución de Lévy y el descarte de nidos menos favorables para la construcción de nuevas soluciones. Además, dado que esta es la última iteración, se obtiene la mejor solución obtenida por el algoritmo. Fuente: Elaboración propia.

Anexo J: Conformidad del especialista en algoritmia para el capítulo 7.

Lima, 25 de septiembre del 2020

ACTA DE CONFORMIDAD

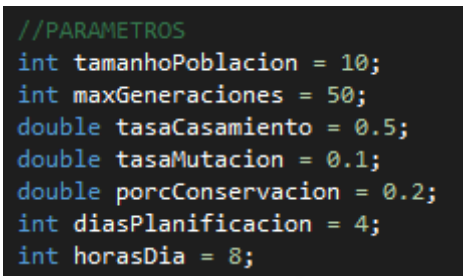
Mediante el presente documento el Sr. CUEVA MOSCOSO RONY, especialista en algoritmia, da conformidad de que el Sr. LABAJOS TRIGOSO STEVEN ALONSO, estudiante del curso de Proyecto de Tesis 2, ha presentado el documento que contiene el diseño del pseudocódigo del algoritmo Cuckoo Search adaptado al problema de planificación. Se han revisado las funcionalidades de actualización de nidos siguiendo una distribución de Lévy y la construcción de nuevas soluciones. La información es adecuada para alcanzar los objetivos esperados.

Ing. Rony Cueva Moscoso

DNI 09942265

Anexo K: Caso de prueba para la ejecución de código del algoritmo genético.

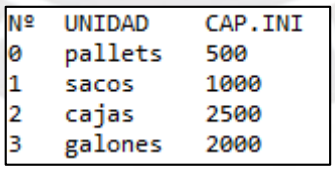
Tabla K1. Parámetros utilizados para la ejecución de prueba del algoritmo genético.

Parámetros

• tamanhoPoblacion: Tamaño de población. • maxGeneraciones: Número máximo de generaciones. • tasaCasamiento: Tasa de casamiento. • tasaMutacion: Tasa de mutación. • porcConservación: Porcentaje de conservación de soluciones. • diasPlanificación: Número de días de planificación. • horasDía: Horas de trabajo por cada dia de planificación.

```
//PARAMETROS
int tamanhoPoblacion = 10;
int maxGeneraciones = 50;
double tasaCasamiento = 0.5;
double tasaMutacion = 0.1;
double porcConservacion = 0.2;
int diasPlanificacion = 4;
int horasDia = 8;
```

Fuente: Elaboración propia.

Tabla K2. Datos aleatorios generados por código para el algoritmo genético.

Almacenes

• Unidad: Unidad de materia prima que se almacena. • Cap. Ini: Capacidad de almacenamiento al inicio de cada día.
Recepciones

Nº	UNIDAD	CAP.INI
0	pallets	500
1	sacos	1000
2	cajas	2500
3	galones	2000

Nº	MP	CANT.	UNIDAD	F.COMPRA	F.IMP.	F.LIMITE
0	MP0	400	pallets	3	-1	3
1	MP1	300	sacos	-1	0	3
2	MP2	200	sacos	-1	2	2
3	MP3	100	pallets	0	-1	3
4	MP4	400	sacos	2	-1	3
5	MP5	200	sacos	-1	2	2
6	MP6	500	sacos	1	-1	1
7	MP7	500	galones	2	-1	3

- MP: Nombre correlativo para la materia prima a recepcionar.
- Cant: Cantidad solicitada de materia prima.
- Unidad: Unidad de la materia prima.

Las fechas se explican, a manera de ejemplo, con la primera recepción N°0:

- F. Compra: Fecha de compra. El número 3 significa que es el día 3 desde el inicio de la planificación.
- F. Imp: Fecha de importación. El valor -1 indica que no hay fecha de importación. Donde se vea este valor, significa que esa fecha no existe.
- F. Límite: Fecha límite de necesidad. El número 3, significa el día 3 desde el inicio de la planificación.

Vehículos

Nº	MP	CANT.TRANSPORTE
0	MP0	200
1	MP0	200
2	MP1	300
3	MP2	100
4	MP2	100
5	MP3	50
6	MP3	50
7	MP4	400
8	MP5	200
9	MP6	250
10	MP6	250
11	MP7	500

- MP: Materia prima asociada a este vehículo.
- Cant.Transporte: Cantidad transportada de la materia prima correspondiente.

Nota: En esta tabla se muestran los resultados que se obtienen de forma aleatoria para el problema y que se toman como base para proceder a realizar la programación de recepciones. Estos datos son importantes ya que son compartidos por todas las posibles soluciones generadas por el algoritmo. Fuente: Elaboración propia.

Tabla K3. Mejores soluciones obtenidas mediante código del algoritmo genético.

Mejor solución inicial

```
========================================
MEJOR SOLUCION INICIAL
CROMOSOMA
      T0   T1   T2   T3   T4   T5   T6   T7
DIA 0 MP1  X    X    X    X    X    X    X
DIA 1 X    MP6  X    MP6  MP3  X    X    X
DIA 2 MP2  MP2  MP5  MP7  MP3  MP4  X    X
DIA 3 X    MP0  MP0  X    X    X    X    X
Fitness: 3.9523809523809526
========================================
```

Mejor solución generación 0

```
========================================
MEJOR SOLUCION GENERACION 0
CROMOSOMA
      T0   T1   T2   T3   T4   T5   T6   T7
DIA 0 MP1  X    X    X    X    X    X    X
DIA 1 X    MP3  X    X    MP6  MP3  X    MP6
DIA 2 MP5  MP2  MP2  MP7  X    X    X    MP4
DIA 3 X    X    X    X    X    X    MP0  MP0
Fitness: 2.05
========================================
```

Mejor solución generación 2

```
========================================
MEJOR SOLUCION GENERACION 2
CROMOSOMA
      T0   T1   T2   T3   T4   T5   T6   T7
DIA 0 MP1  X    X    X    X    X    X    X
DIA 1 X    MP6  X    MP6  MP3  X    X    X
DIA 2 MP2  MP2  MP5  MP7  MP3  MP4  X    X
DIA 3 X    MP0  MP0  X    X    X    X    X
Fitness: 1.8833333333333333
========================================
```

Mejor solución generación 7

```
========================================
MEJOR SOLUCION GENERACION 7
CROMOSOMA
      T0   T1   T2   T3   T4   T5   T6   T7
DIA 0 MP1  X    X    X    X    X    X    X
DIA 1 X    MP6  X    MP6  MP3  X    X    X
DIA 2 MP2  MP2  MP5  MP7  X    MP4  X    X
DIA 3 X    MP0  MP0  X    X    X    MP3  X
Fitness: 1.7722222222222221
========================================
```

Mejor solución generación 12

```
=======================================
MEJOR SOLUCION GENERACION 12
CROMOSOMA
      T0   T1   T2   T3   T4   T5   T6   T7
DIA 0 MP1  X    X    X    X    X    X    X
DIA 1 X    MP6  X    MP6  MP3  X    X    X
DIA 2 MP2  MP2  MP5  MP7  X    X    X    X
DIA 3 X    MP0  MP0  MP4  X    X    MP3  X
Fitness: 1.3722222222222222
=======================================
```

Mejor solución generación 19

```
=======================================
MEJOR SOLUCION GENERACION 19
CROMOSOMA
      T0   T1   T2   T3   T4   T5   T6   T7
DIA 0 MP1  X    X    MP3  X    X    X    X
DIA 1 X    MP6  X    MP6  X    X    X    X
DIA 2 MP2  MP2  MP5  MP3  X    X    X    X
DIA 3 MP7  MP0  MP0  MP4  X    X    X    X
Fitness: 1.3444444444444446
=======================================
```

Mejor solución generación 22

```
=======================================
MEJOR SOLUCION GENERACION 22
CROMOSOMA
      T0   T1   T2   T3   T4   T5   T6   T7
DIA 0 MP1  X    X    X    X    X    X    X
DIA 1 X    MP6  X    MP6  MP3  X    X    X
DIA 2 MP2  MP2  MP5  MP3  X    X    X    X
DIA 3 MP7  MP0  MP0  MP4  X    X    X    X
Fitness: 1.2333333333333334
=======================================
```

Mejor solución generación 29

```
=======================================
MEJOR SOLUCION GENERACION 29
CROMOSOMA
      T0   T1   T2   T3   T4   T5   T6   T7
DIA 0 MP1  X    X    X    X    X    X    X
DIA 1 X    MP6  X    MP6  MP3  X    X    X
DIA 2 MP2  MP2  MP5  X    X    X    X    X
DIA 3 MP7  MP0  MP0  MP4  MP3  X    X    X
Fitness: 1.1222222222222222
=======================================
```

Mejor solución generación 34

```
=======================================
MEJOR SOLUCION GENERACION 34
CROMOSOMA
      T0   T1   T2   T3   T4   T5   T6   T7
DIA 0 MP1  X    X    X    X    X    X    X
DIA 1 X    MP6  X    MP6  X    X    X    X
DIA 2 MP2  MP2  MP5  X    MP3  X    X    X
DIA 3 MP7  MP0  MP0  MP4  MP3  X    X    X
Fitness: 1.011111111111111
=======================================
```

Mejor solución final

```
=======================================
MEJOR SOLUCION FINAL
CROMOSOMA
      T0   T1   T2   T3   T4   T5   T6   T7
DIA 0 MP1  X    X    X    X    X    X    X
DIA 1 X    MP6  X    MP6  X    X    X    X
DIA 2 MP2  MP2  MP5  X    MP3  X    X    X
DIA 3 MP7  MP0  MP0  MP4  MP3  X    X    X
Fitness: 1.011111111111111
=======================================
```

- T0 – T7: Horas de trabajo, en este ejemplo, 8 horas.
- X: Ninguna recepción se asignó en este espacio de tiempo.
- MPX: La materia prima X se asignó en este espacio de tiempo. Ejemplo: Recepción de MP1 (Día 0,T0).

Nota: En esta tabla se presentan las soluciones que mejoraron respecto a las generaciones anteriores. Se puede observar que la mejor solución encontrada para el problema ocurrió en la generación 34. Fuente: Elaboración propia.

Explicación de la mejor solución encontrada:

```
=======================================
MEJOR SOLUCION FINAL
CROMOSOMA
      T0   T1   T2   T3   T4   T5   T6   T7
DIA 0 MP1  X    X    X    X    X    X    X
DIA 1 X    MP6  X    MP6  X    X    X    X
DIA 2 MP2  MP2  MP5  X    MP3  X    X    X
DIA 3 MP7  MP0  MP0  MP4  MP3  X    X    X
Fitness: 1.011111111111111
=======================================
```

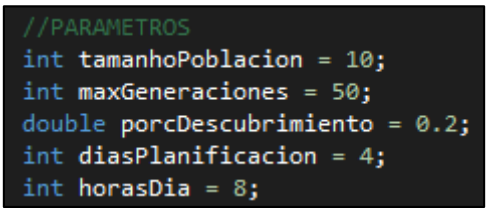
Para ejemplificar lo que hace el algoritmo, en la solución final, la MP3 se podía recibir en cualquiera de los días de planificación (Fecha de compra: 0, Fecha límite: 3), y lo que fue buscando el proceso de optimización es que esté lo más cerca posible a la fecha límite, colocando, en este caso, una recepción en el día 2 y otra en el día 3.

Si se verifica la capacidad utilizada de almacenamiento para la unidad "pallets", asociada a la MP3, en el día 3, se calcula que su uso fue de 450 pallets, compuesto por dos recepciones de 200 pallets de MP0 y una de 50 pallets de MP3. Si se habría definido un número mayor de generaciones, es posible que el algoritmo hubiese generado una solución con las dos recepciones de MP3 en el día 3, situación ideal debido a lo comentado y dado que la capacidad disponible de almacenamiento diaria lo permite.

Esto es lo que busca hacer el algoritmo con cada una de las recepciones, programarla para los días más cercanos a su fecha límite de necesidad, tratando, a su vez, que se reciba la mayor cantidad posible de la misma, de tal forma que su tiempo de almacenamiento sea menor.

Anexo L: Caso de prueba para la ejecución de código del algoritmo Cuckoo Search.

Tabla L1. Parámetros utilizados para la ejecución de prueba del algoritmo Cuckoo Search.

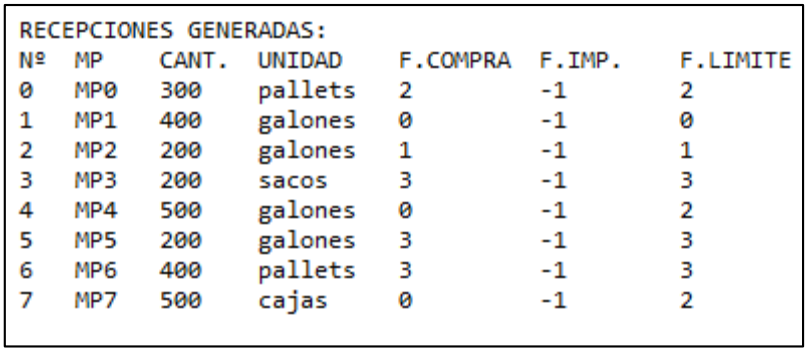
Parámetros

• tamanhoPoblacion: Tamaño de población. • maxGeneraciones: Número máximo de generaciones. • porcDescubrimiento: Porcentaje de descubrimiento para peores nidos. • diasPlanificación: Número de días de planificación. • horasDía: Horas de trabajo por cada dia de planificación.

```
//PARAMETROS
int tamanhoPoblacion = 10;
int maxGeneraciones = 50;
double porcDescubrimiento = 0.2;
int diasPlanificacion = 4;
int horasDia = 8;
```

Fuente: Elaboración propia.

Tabla L2. Datos aleatorios generados por código para el algoritmo Cuckoo Search.

Almacenes

• Unidad: Unidad de materia prima que se almacena. • Cap. Ini: Capacidad de almacenamiento al inicio de cada día.
Recepciones


Nº	UNIDAD	CAP.INI
0	pallets	1000
1	sacos	2500
2	cajas	500
3	galones	3000

RECEPCIONES GENERADAS:

Nº	MP	CANT.	UNIDAD	F.COMPRA	F.IMP.	F.LIMITE
0	MP0	300	pallets	2	-1	2
1	MP1	400	galones	0	-1	0
2	MP2	200	galones	1	-1	1
3	MP3	200	sacos	3	-1	3
4	MP4	500	galones	0	-1	2
5	MP5	200	galones	3	-1	3
6	MP6	400	pallets	3	-1	3
7	MP7	500	cajas	0	-1	2

- MP: Nombre correlativo para la materia prima a recepcionar.
- Cant: Cantidad solicitada de materia prima.
- Unidad: Unidad de la materia prima.

Las fechas se explican, a manera de ejemplo, con la primera recepción N°0:

- F. Compra: Fecha de compra. El número 2 significa que es el día 2 desde el inicio de la planificación.
- F. Imp: Fecha de importación. El valor -1 indica que no hay fecha de importación. Donde se vea este valor, significa que esa fecha no existe.
- F. Límite: Fecha límite de necesidad. El número 2, significa el día 2 desde el inicio de la planificación.

Vehículos

VEHICULOS DE TRANSPORTE:

Nº	MP	CANT.TRANSPORTE
0	MP0	300
1	MP1	400
2	MP2	100
3	MP2	100
4	MP3	100
5	MP3	100
6	MP4	250
7	MP4	250
8	MP5	200
9	MP6	200
10	MP6	200
11	MP7	250
12	MP7	250

- MP: Materia prima asociada a este vehículo.
- Cant.Transporte: Cantidad transportada de la materia prima correspondiente.

Nota: En esta tabla se muestran los resultados que se obtienen de forma aleatoria para el problema y que se toman como base para proceder a realizar la programación de recepciones. Estos datos son importantes ya que son compartidos por todas las posibles soluciones generadas por el algoritmo. Fuente: Elaboración propia.

Tabla L3. Mejores soluciones obtenidas mediante código del algoritmo genético.

Mejor solución inicial

```
=======================================
MEJOR SOLUCION INICIAL
      T0   T1   T2   T3   T4   T5   T6   T7
DIA 0 MP1  MP7  X    X    X    X    X    X
DIA 1 MP2  MP2  X    X    X    MP4  MP4  MP7
DIA 2 X    X    X    X    MP0  X    X    X
DIA 3 X    MP6  MP6  MP3  MP5  MP3  X    X
Fitness: 3.217391304347826
=======================================
```

Mejor solución generación 5

```
=======================================
MEJOR SOLUCION GENERACION 5
      T0   T1   T2   T3   T4   T5   T6   T7
DIA 0 MP1  MP4  X    X    X    MP7  X    MP4
DIA 1 MP2  MP2  X    X    X    X    X    X
DIA 2 X    X    MP7  X    X    X    X    MP0
DIA 3 X    MP6  X    MP6  MP3  X    MP3  MP5
Fitness: 2.4761904761904763
=======================================
```

Mejor solución generación 24

```
=======================================
MEJOR SOLUCION GENERACION 24
      T0   T1   T2   T3   T4   T5   T6   T7
DIA 0 X    X    X    MP4  MP7  MP1  X    X
DIA 1 MP2  MP2  X    X    X    MP4  X    X
DIA 2 X    MP7  X    X    X    X    MP0  X
DIA 3 X    X    MP6  MP3  X    MP6  MP3  MP5
Fitness: 2.310805173133083
=======================================
```

Mejor solución final

```
=======================================
MEJOR SOLUCION FINAL
      T0   T1   T2   T3   T4   T5   T6   T7
DIA 0 X    X    X    MP4  MP7  MP1  X    X
DIA 1 MP2  MP2  X    X    X    MP4  X    X
DIA 2 X    MP7  X    X    X    X    MP0  X
DIA 3 X    X    MP6  MP3  X    MP6  MP3  MP5
Fitness: 2.310805173133083
=======================================
```

- T0 – T7: Horas de trabajo, en este ejemplo, 8 horas.
- X: Ninguna recepción se asignó en este espacio de tiempo.
- MPX: La materia prima X se asignó en este espacio de tiempo. Ejemplo: Recepción de MP1 (Día 0,T0).

Nota: En esta tabla se presentan las soluciones que mejoraron respecto a las generaciones anteriores. Se puede observar que la mejor solución encontrada para el problema ocurrió en la generación 24. Fuente: Elaboración propia.

Explicación de la mejor solución encontrada:

```
========================================
MEJOR SOLUCION FINAL
       T0   T1   T2   T3   T4   T5   T6   T7
DIA 0  X    X    X    MP4  MP7  MP1  X    X
DIA 1  MP2  MP2  X    X    X    MP4  X    X
DIA 2  X    MP7  X    X    X    X    MP0  X
DIA 3  X    X    MP6  MP3  X    MP6  MP3  MP5
Fitness: 2.310805173133083
========================================
```

Para ejemplificar lo que hace el algoritmo, en la solución final, la MP4 se podía recibir en cualquiera de los días de planificación entre el 0 y el 2. Lo que fue buscando el proceso de optimización es que esté lo más cerca posible a la fecha límite, colocando, en este caso, una recepción en el día 0 y otra en el día 1. Nótese que, en la mejor solución anterior a esta última generada, ambas recepciones estaban en el día 0 solamente.

Al igual que para el caso de prueba del algoritmo genético, si se habría definido un número mayor de iteraciones, es posible que el algoritmo hubiese generado una solución con las dos recepciones de MP4 en el día 2, situación ideal debido a lo comentado.

Se comprueba entonces que el algoritmo Cuckoo Search está buscando mejorar las planificaciones realizadas, programando las recepciones para los días más cercanos a su fecha límite de necesidad, tratando, a su vez, que se reciba la mayor cantidad posible de la misma, de tal forma que su tiempo de almacenamiento sea menor.

Anexo M: Prueba de integración entre pantallas y algoritmos.

Archivos utilizados para la prueba de integración en ambos algoritmos:

- **Almacenes.csv:**

  El contenido de cada una de las líneas es el siguiente:

  - Unidad de almacén.
  - Capacidad inicial diaria.

```
1 pallets,500
2 sacos,1000
3 cajas,2500
4 galones,2000
```

- **Solicitudes.csv:**

  Este archivo tiene los siguientes datos:

  - Línea 1: días de planificación, horas de trabajo diarias.
  - Línea 2: Fecha inicial de planificación.
  - Línea 3: Hora inicial de trabajo diaria.

  A continuación, cada fila contiene lo siguiente:

  - Nombre de la materia prima
  - Cantidad solicitada
  - Unidad de materia prima
  - Fecha de compra
  - Fecha de importación
  - Fecha límite de necesidad.

```
4,8
10/10/2020
9
MP0,400,pallets,3,-1,3
MP1,300,sacos,-1,0,3
MP2,200,sacos,-1,2,2
MP3,100,pallets,0,-1,3
MP4,400,sacos,2,-1,3
MP5,200,sacos,-1,2,2
MP6,500,sacos,1,-1,1
MP7,500,galones,2,-1,3
```

- **Vehiculos.csv:**

  El contenido de cada una de las líneas es el siguiente:

  - Placa de vehículo
  - Materia prima transportada
  - Cantidad transportada de la materia prima asociada.

```
AXD-066,MP0,200
HDB-732,MP0,200
KMS-923,MP1,300
OLS-425,MP2,100
IXB-807,MP2,100
EYS-567,MP3,50
OAL-607,MP3,50
HQQ-139,MP4,400
PLM-062,MP5,200
XNA-200,MP6,250
DAS-565,MP6,250
XMA-983,MP7,500
```

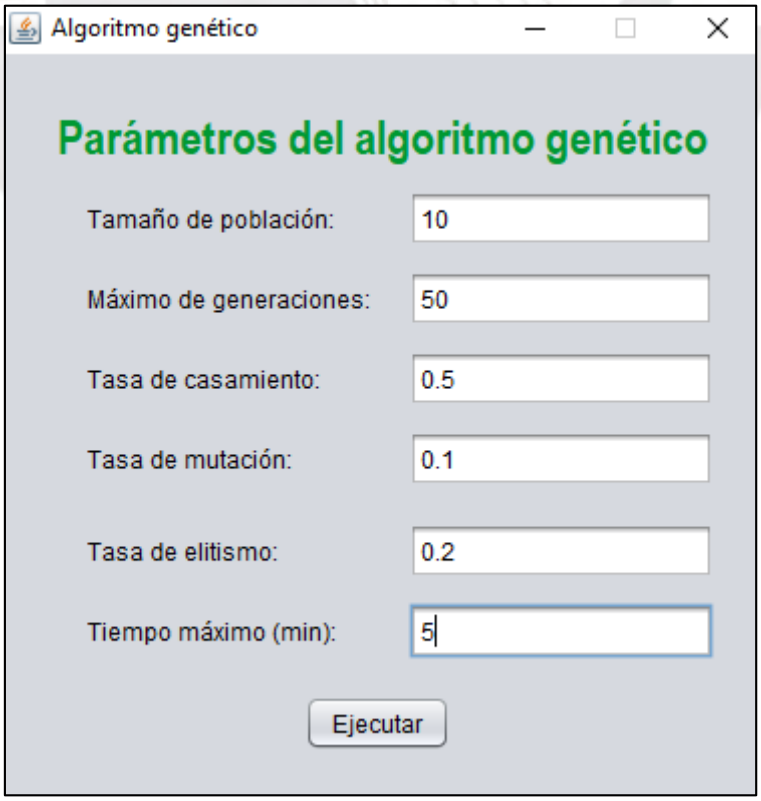
Prueba de integración con el algoritmo genético:

- Pantalla 1

Se cargan los archivos de datos correspondientes y se selecciona el algoritmo genético.

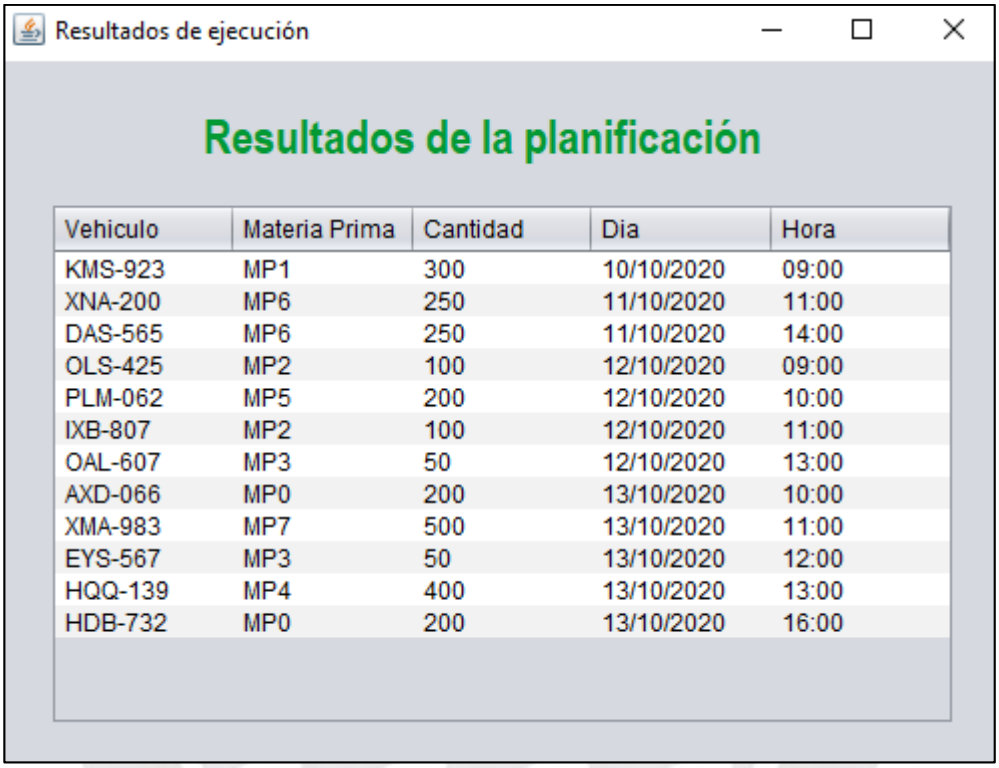
- Pantalla 2

Se ingresan los parámetros del algoritmo genético deseados para la ejecución. Luego, se presiona el botón "Ejecutar".



- Pantalla 3

Se presentan los resultados obtenidos para la planificación de citas de recepción de materia prima según los datos ingresados y la configuración del algoritmo.



Vehiculo	Materia Prima	Cantidad	Dia	Hora
KMS-923	MP1	300	10/10/2020	09:00
XNA-200	MP6	250	11/10/2020	11:00
DAS-565	MP6	250	11/10/2020	14:00
OLS-425	MP2	100	12/10/2020	09:00
PLM-062	MP5	200	12/10/2020	10:00
IXB-807	MP2	100	12/10/2020	11:00
OAL-607	MP3	50	12/10/2020	13:00
AXD-066	MP0	200	13/10/2020	10:00
XMA-983	MP7	500	13/10/2020	11:00
EYS-567	MP3	50	13/10/2020	12:00
HQQ-139	MP4	400	13/10/2020	13:00
HDB-732	MP0	200	13/10/2020	16:00

Se puede comprobar que esta visualización de resultados obtiene los datos correctamente, ya que, a continuación, se muestra la planificación resultante directamente desde el IDE luego de la ejecución del algoritmo genético:

```
========================================
MEJOR SOLUCION FINAL
       T0   T1   T2   T3   T4   T5   T6   T7
DIA 0  MP1  X    X    X    X    X    X    X
DIA 1  X    X    MP6  X    X    MP6  X    X
DIA 2  MP2  MP5  MP2  X    MP3  X    X    X
DIA 3  X    MP0  MP7  MP3  MP4  X    X    MP0
Fitness: 1.396825396825397
========================================
```

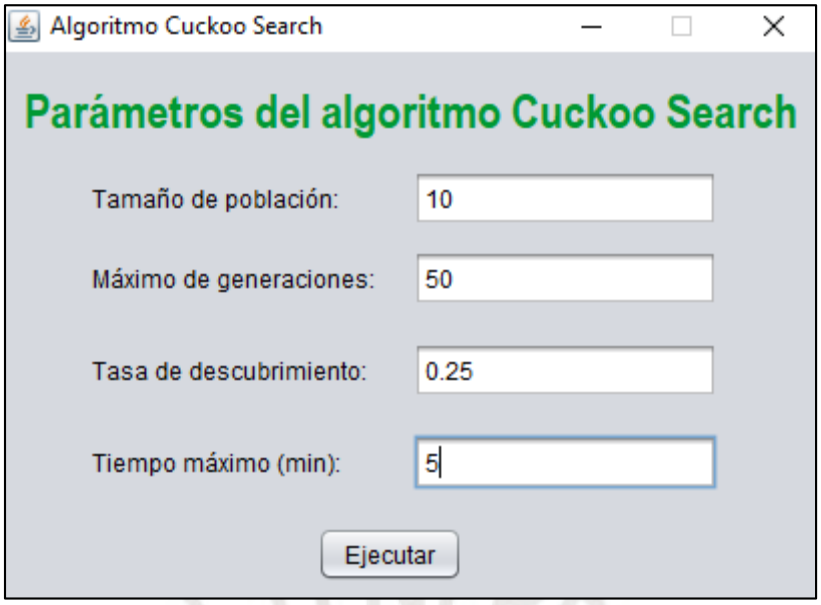
Prueba de integración con el algoritmo Cuckoo Search:

- Pantalla 1

Se cargan los archivos de datos correspondientes y se selecciona el algoritmo Cuckoo Search.

- Pantalla 2

Se ingresan los parámetros del algoritmo Cuckoo Search deseados para la ejecución. Luego, se presiona el botón "Ejecutar".



- Pantalla 3

Se presentan los resultados obtenidos para la planificación de citas de recepción de materia prima según los datos ingresados y la configuración del algoritmo.

Vehiculo	Materia Prima	Cantidad	Dia	Hora
KMS-923	MP1	300	10/10/2020	09:00
XNA-200	MP6	250	11/10/2020	09:00
DAS-565	MP6	250	11/10/2020	13:00
PLM-062	MP5	200	12/10/2020	09:00
OLS-425	MP2	100	12/10/2020	10:00
IXB-807	MP2	100	12/10/2020	11:00
EYS-567	MP3	50	12/10/2020	12:00
XMA-983	MP7	500	12/10/2020	14:00
OAL-607	MP3	50	12/10/2020	16:00
HQQ-139	MP4	400	13/10/2020	09:00
AXD-066	MP0	200	13/10/2020	13:00
HDB-732	MP0	200	13/10/2020	14:00

Se puede comprobar que esta visualización de resultados obtiene los datos correctamente, ya que, a continuación, se muestra la planificación resultante directamente desde el IDE luego de la ejecución del algoritmo Cuckoo Search:

```
==========================================
MEJOR SOLUCION FINAL
       T0   T1   T2   T3   T4   T5   T6   T7
DIA 0  MP1  X    X    X    X    X    X    X
DIA 1  MP6  X    X    X    MP6  X    X    X
DIA 2  MP5  MP2  MP2  MP3  X    MP7  X    MP3
DIA 3  MP4  X    X    X    MP0  MP0  X    X
Fitness: 1.869047619047619
==========================================
```

Anexo N: Direcciones URL para los algoritmos y la interfaz.

- Algoritmo Cuckoo Search

https://drive.google.com/file/d/1BRmCk5_5uyP7OK42Ib7ttFPu_T_wbZSf/view?usp=sharing

- Algoritmo genético

https://drive.google.com/file/d/1eiwSZ7etOSun3qlT4SZESCDX3-iHUZWu/view?usp=sharing

- Interfaz gráfica

https://drive.google.com/file/d/1Wi0O2VVcPmznkhXlhzI7EBgQnGwImRjJ/view?usp=sharing

Anexo O: Conformidad del especialista en algoritmia para el capítulo 11.

Lima, 23 de octubre del 2020

ACTA DE CONFORMIDAD

Mediante el presente documento el Sr. CUEVA MOSCOSO RONY, especialista en algoritmia, da conformidad de que el Sr. LABAJOS TRIGOSO STEVEN ALONSO, estudiante del curso de Proyecto de Tesis 2, ha presentado el documento que contiene la calibración de variables para el algoritmo Cuckoo Search, considerando como punto de partida los valores sugeridos por la literatura y revisando el valor de la función objetivo en cada modificación realizada a los parámetros del mismo. La información es adecuada para alcanzar los objetivos esperados.

Ing. Rony Cueva Moscoso

DNI 09942265

Anexo P: Valores fitness utilizados para la experimentación numérica.

Tabla P1. Valores fitness para experimentación numérica.

Muestra	Fitness de algoritmo genético	Fitness de algoritmo Cuckoo Search
1	10.6111	12.4444
2	15.8806	17.6944
3	13.1583	17.6056
4	6.6056	8.2667
5	6.7944	8.9278
6	10.6667	12.5000
7	7.5056	10.4750
8	11.3722	12.5500
9	9.7833	11.6500
10	13.1611	15.3278
11	13.1611	15.3278
12	13.2944	17.7306
13	15.8611	17.4972
14	10.7222	12.4333
15	13.2778	15.3778
16	9.8611	11.6889
17	7.6917	10.0583
18	6.7222	8.1222

19	9.7028	12.5361
20	5.4972	7.3778
21	13.1528	17.6639
22	9.8056	11.5222
23	13.1222	15.4111
24	7.6167	10.3500
25	9.8056	11.5722
26	9.9861	12.2278
27	15.8694	17.5556
28	9.8167	11.5778
29	7.6167	10.3500
30	7.7917	10.2417
31	9.9250	12.2972
32	9.9861	12.2278
33	7.7556	10.5167
34	11.3583	12.9250
35	11.5500	12.7778
36	9.6583	12.5833
37	5.4694	7.4000
38	9.3028	10.5528
39	11.5056	12.9389

40	6.8167	8.9944

Fuente: Elaboración propia.

Anexo Q: Conformidad del especialista en algoritmia para el capítulo 12.

Lima, 30 de octubre del 2020

ACTA DE CONFORMIDAD

Mediante el presente documento el Sr. CUEVA MOSCOSO RONY, especialista en algoritmia, da conformidad de que el Sr. LABAJOS TRIGOSO STEVEN ALONSO, estudiante del curso de Proyecto de Tesis 2, ha presentado el informe de experimentación numérica, el cual contiene la recolección de datos mediante diversos casos de prueba y el análisis para evaluar el rendimiento de los algoritmos trabajados utilizando las pruebas estadísticas correspondientes. La información es adecuada para alcanzar los objetivos esperados.

Ing. Rony Cueva Moscoso

DNI 09942265